

AD-A267 494 JMENTATION PAGE

Form Approved
OMB No. 0704-0188

It is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including this burden estimate, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

2. REPORT DATE MAY 1993		3. REPORT TYPE AND DATES COVERED THESIS/DISSERTATION	
4. TITLE AND SUBTITLE State Space Partitioning Methods for Solving a Class of Stochastic Network Problems		5. FUNDING NUMBERS	
6. AUTHOR(S) Jay Alan Jacobson		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/CI/CIA- 93-006D	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFIT Student Attending: Georgia Institute of Technology		10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) DEPARTMENT OF THE AIR FORCE AFIT/CI 2950 P STREET WRIGHT-PATTERSON AFB OH 45433-7765		11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public Release IAW 190-1 Distribution Unlimited MICHAEL M. BRICKER, SMSgt, USAF Chief Administration		12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words)			
14. SUBJECT TERMS		15. NUMBER OF PAGES 123	
17. SECURITY CLASSIFICATION OF REPORT		16. PRICE CODE	
18. SECURITY CLASSIFICATION OF THIS PAGE		19. SECURITY CLASSIFICATION OF ABSTRACT	
		20. LIMITATION OF ABSTRACT	

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std. Z39-18
298-102

93-006D

State Space Partitioning Methods for Solving a Class of Stochastic Network Problems

A THESIS
Presented to
The Academic Faculty

by

Jay Alan Jacobson

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Industrial and Systems Engineering

Georgia Institute of Technology
May 10, 1993

Copyright ©1993 by Jay Alan Jacobson

DTIC QUALITY INSPECTED 3

Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

State Space Partitioning Methods
for Solving a
Class of Stochastic Network Problems

APPROVED:

Christos Alexopoulos

Christos Alexopoulos, Chairman

David M. Goldsman

David M. Goldsman

Donna Crystal Llewellyn

Donna Crystal Llewellyn

Craig A. Tovey

Craig A. Tovey

Yorai Wardi

Yorai Wardi

Date Approved by Chairman May 10, 1993

DEDICATION

This thesis is lovingly dedicated to Almighty God,
the Source of all that I am,
through Whom all things are possible;
to His Son Jesus Christ, the Hope of my salvation; and
to the Holy Spirit, Whose peace and strength sustain me.

All glory and honor be to the Holy Trinity,
now and forever.

Amen.

ACKNOWLEDGEMENTS

I am grateful to my thesis advisor, Dr. Christos Alexopoulos. His knowledge of the frontier in this area helped me to quickly identify my topic, and his limitless patience helped me to progress rapidly. I truly could not have accomplished this without him.

Thanks also to the other members of my reading committee: Dr. David Goldman; Dr. Donna Llewellyn; Dr. Craig Tovey; and Dr. Yorai Wardi. Their guidance and suggestions led to major improvements in the thesis. Special thanks to Dr. Tovey, whose observations led me to develop the theory of Section 2.5.

I am grateful to the Department of Mathematical Sciences at the United States Air Force Academy for sponsoring my masters and doctoral degree programs.

Thanks to my parents, James and Diana Jacobson, for giving me an early love of learning and for teaching me how to work and persevere.

Lastly, and most importantly, I am grateful to my wife, Jennifer, and my sons, John and William. I cannot thank them enough for their loving support and their sacrifices.

Contents

Dedication	iii
Acknowledgements	iv
List of Tables	vii
List of Figures	ix
Summary	x
1 Introduction	1
2 General State Space Decomposition Algorithm	11
2.1 Introduction	11
2.2 Definitions and Preliminaries	13
2.3 General Problem Statement	18
2.4 The GSD Algorithm	18
2.5 Theoretical Results Concerning GSD	28
2.6 Using GSD to Produce Bounds	34
2.7 GSD Bounds as Input to Monte Carlo Estimation Routines	36
2.8 Conclusions	37
3 The Stochastic Minimum Spanning Tree Problem	39
3.1 Introduction	39
3.2 Computing the Probability Distribution of the Weight of a MST	42
3.2.1 Spanning Tree Decomposition Algorithm I	43

3.2.2	Spanning Tree Decomposition Algorithm II	47
3.2.3	Hybrid Spanning Tree Decomposition Algorithm	51
3.2.4	Computing the Entire Distribution of the Minimum Spanning Tree Weight	52
3.3	Computing Criticality Indices for the Stochastic MST Problem	54
3.4	Stochastic MST Examples	62
3.5	Extension of MST Results to Other Stochastic Matroid Settings . . .	72
3.6	Conclusions	73
4	The Stochastic Minimum Cost Network Flow Problem	75
4.1	Introduction	75
4.2	The Case of Independent Costs and Capacities	77
4.2.1	Decomposition Algorithms I	78
4.2.2	Decomposition Algorithms II	81
4.2.3	GSD Routines for the Independent Case	83
4.2.4	Hybrid Decomposition Routine for the Independent Case . . .	86
4.2.5	Computing the Entire Distribution of MCF Costs	87
4.3	The Case of Dependent Costs and Capacities	89
4.3.1	Decomposition Algorithms I	90
4.3.2	Decomposition Algorithms II	92
4.3.3	GSD Routines for the Dependent Case	93
4.3.4	Hybrid Decomposition Routine for the Dependent Case	95
4.3.5	Computing the Entire Distribution of MCF Costs	96
4.4	Stochastic MCF Examples	96
4.5	Conclusions	110
5	Conclusions and Directions for Further Study	116
	Bibliography	119
	Vita	123

List of Tables

1.1	Input Distribution for Example 1.1 and Example 1.2.	2
1.2	Arc Cost Distributions for Example 1.3.	4
1.3	Arc Capacity Distributions for Example 1.3.	5
1.4	Sample of decomposition results for Examples 1.1 and 1.2.	10
1.5	Sample of decomposition results for Example 1.3.	10
3.1	Input Distribution for Example 3.1.	64
3.2	Computing $F(60)$ for Example 3.1.	65
3.3	Computing $F(90)$ for Example 3.1.	65
3.4	Partial listing of the cdf $F(d)$ for Example 3.1.	66
3.5	Criticality indices for Example 3.1.	67
3.6	Input Distribution for Example 3.2.	68
3.7	Computing $F(280)$ for Example 3.2.	69
3.8	Computing $F(400)$ for Example 3.2.	69
3.9	Partial listing of the cdf $F(d)$ for Example 3.2.	70
3.10	Criticality indices for Example 3.2.	71
4.1	Arc Cost Distributions for Example 4.2.	98
4.2	Arc Capacity Distributions for Example 4.2.	99
4.3	Computing $F(1900)$ for Example 4.2.	100
4.4	Computing $F(3500)$ for Example 4.2.	101
4.5	Arc Cost Distributions for Example 4.3.	102
4.6	Arc Capacity Distributions for Example 4.3.	103
4.7	Computing $F(1800)$ for Example 4.3.	104
4.8	Bounding $F(2000)$ for Example 4.3.	104

4.9	Bounding the cdf $F(d)$ for Example 4.3.	105
4.10	Input Probability Distribution for Example 4.4.	106
4.11	Computing $F(1900)$ for Example 4.4.	107
4.12	Computing $F(3000)$ for Example 4.4.	108
4.13	Partial listing of the cdf $F(d)$ for Example 4.4.	109
4.14	Input Probability Distribution for Example 4.5.	111
4.15	Computing $F(1650)$ for Example 4.5.	112
4.16	Computing $F(2000)$ for Example 4.5.	113
4.17	Partial listing of cdf $F(d)$ for Example 4.5.	114

List of Figures

2.1 Discrete interval given in Example 2.1.	14
---	----

SUMMARY

This study focuses on state space partitioning techniques for computing measures related to the operation of stochastic systems. These methods iteratively decompose the system state space until the measure of interest has been determined. The information available in each iteration yields lower and upper bounds on this measure, and can be used to design efficient Monte Carlo estimation routines. We present here new theoretical results identifying strategies for significantly enhancing the performance of these algorithms. Using these results, we describe a generalized algorithm that can easily be tailored to address a variety of problems. We next use this algorithm to analyze two important models in the area of stochastic network optimization. The first concerns the probabilistic behavior of minimum spanning trees in graphs with discrete random arc weights. Specifically, we compute the probability distribution of the weight of a minimum spanning tree and the probability that a given arc is on a minimum spanning tree. Both of these problems are shown to be #P-hard. The second model considers minimum cost flows in networks with discrete random arc costs and capacities. We consider the case of statistically independent costs and capacities for each arc as well as the case in which the cost and capacity of each arc change simultaneously. In each case, we show that the evaluation of the distribution of the minimum cost flow for a fixed demand configuration is a #P-hard problem. Numerical examples are given throughout the thesis.

CHAPTER 1

Introduction

Much work has been done on the various problems that arise in the area of network optimization, resulting in a rich and elegant theory. We assume familiarity with this theory, and will use its terms and results without lengthy explanation. See [24],[28],[29] for an overview of network theory. Network optimization problems are unique among integer and combinatorial optimization problems in that efficient algorithms for their solution generally exist. These tools can solve an extensive array of optimization problems, including many that do not naturally involve networks but can be modeled using networks.

In practice, unfortunately, the situation is complicated by the fact that some of the network parameters (e.g., arc weights or capacities) either are not known with certainty or change in some probabilistic fashion. In such cases, it would likely be necessary to make multiple optimization runs using a number of possible values for those parameters. In fact, in order to best gain insight into an uncertain situation one might assign probabilities to many possible parameter settings and solve each resulting optimization problem separately. The resulting distribution would then enable one to make probabilistic statements about the value of an optimal solution.

Example 1.1 Suppose a communication network is to be built to connect the nodes in an undirected graph, using a least-weight subset of possible arcs (*weight* may represent cost, amount of material, etc.). During the design phase, the weights of these arcs are not known with certainty; rather, they are independent discrete

Table 1.1: Input Distribution for Example 1.1 and Example 1.2.

Arc	Weight (Probability)		
1 = (1,2)	2.0 (0.90)	8.0 (0.08)	12.0 (0.02)
2 = (1,3)	10.0 (0.85)	24.0 (0.12)	35.0 (0.03)
3 = (1,4)	6.0 (0.88)	18.0 (0.10)	24.0 (0.02)
4 = (2,5)	17.0 (0.75)	35.0 (0.20)	50.0 (0.05)
5 = (2,6)	3.0 (0.68)	7.0 (0.25)	10.0 (0.07)
6 = (2,3)	12.0 (0.85)	22.0 (0.11)	30.0 (0.04)
7 = (3,7)	10.0 (0.80)	19.0 (0.14)	24.0 (0.06)
8 = (3,8)	4.0 (0.75)	6.0 (0.17)	10.0 (0.08)
9 = (3,4)	3.0 (0.84)	7.0 (0.13)	12.0 (0.03)
10 = (4,9)	21.0 (0.85)	33.0 (0.09)	45.0 (0.06)
11 = (5,7)	8.0 (0.77)	14.0 (0.13)	18.0 (0.10)
12 = (5,10)	15.0 (0.76)	21.0 (0.22)	25.0 (0.02)
13 = (3,6)	5.0 (0.65)	10.0 (0.23)	12.0 (0.12)
14 = (5,6)	18.0 (0.94)	27.0 (0.05)	36.0 (0.01)
15 = (6,7)	25.0 (0.80)	38.0 (0.12)	50.0 (0.08)
16 = (7,8)	20.0 (0.87)	30.0 (0.09)	40.0 (0.04)
17 = (7,10)	4.0 (0.75)	9.0 (0.14)	15.0 (0.11)
18 = (4,8)	9.0 (0.82)	13.0 (0.15)	20.0 (0.03)
19 = (8,9)	15.0 (0.79)	28.0 (0.15)	45.0 (0.06)
20 = (7,9)	10.0 (0.95)	17.0 (0.03)	30.0 (0.02)
21 = (9,10)	8.0 (0.85)	14.0 (0.10)	25.0 (0.05)

random variables with the probability distributions given in Table 1.1. (For example, arc 1 will have weight 2.0 with probability 0.9, 8.0 with probability 0.08, etc.) This is a stochastic version of the classic minimum spanning tree (MST) problem. If we are constrained to build the MST at a weight that does not exceed a given value d , we might like to know the probability that we will be successful. In fact, it would be helpful to determine the entire distribution of the weight of a MST so that the expected weight and higher moments may be computed.

Example 1.2 Given the same network and arc weight distribution, we are interested in computing criticality indices of arcs, as defined below.

Definition 1.1 The *criticality index* for arc e is the probability that this arc belongs to a minimum spanning tree.

The arcs with the largest criticality indices are most likely to be on the least-weight spanning tree described above.

Example 1.3 In a directed flow network of 10 nodes and 21 arcs with source $s = 1$, sink $t = 10$, and arcs having independent random costs and capacities per unit of flow transmitted given in Tables 1.2 and 1.3, respectively, we wish to derive the distribution of the cost of a minimum cost $s-t$ flow satisfying a demand requirement for 15 units at node 10.

Note that uncertainty about network parameters can come about in a number of ways. In the examples above, we had the type of uncertainty that is necessarily involved in planning. In some systems the random variation can be approximated by discrete distributions, as in the following examples.

Example 1.4 Data packets of a uniform size are transmitted over the links of a communication network. Each packet contains information represented in bits, each of which is subject to transmission error (with probability that may depend on the link being used). From this we derive the probability that a packet arrives at the other end of a given link containing errors. If an arriving packet contains an error, it must be re-transmitted. Thus the amount of time it takes for a packet to be correctly transmitted across a link depends on the length of the link and on the total number of transmissions needed for successful transmission (a geometric random variable if we assume that the several transmissions of a packet constitute independent Bernoulli trials with constant probability of success), which in turn depends on the particular bit error characteristics of that link. Then the resulting transmission time distribution is naturally discrete and, from a practical standpoint, can be truncated and thus be considered to have finite state space. If the network is to be connected using only the most economical links, we are interested in investigating

Table 1.2: Arc Cost Distributions for Example 1.3.

Arc	Arc Cost (Probability)			
1 = (1,2)	70 (0.5)	73 (0.3)	94 (0.2)	
2 = (1,3)	25 (0.5)	35 (0.4)	82 (0.1)	
3 = (1,4)	42 (0.5)	48 (0.3)	61 (0.2)	
4 = (2,5)	26 (0.4)	31 (0.3)	55 (0.2)	88 (0.1)
5 = (2,6)	58 (0.4)	70 (0.3)	95 (0.3)	
6 = (3,2)	15 (0.6)	73 (0.4)		
7 = (3,7)	65 (0.5)	74 (0.4)	75 (0.1)	
8 = (3,8)	59 (0.6)	72 (0.3)	98 (0.1)	
9 = (4,3)	21 (0.3)	32 (0.3)	85 (0.2)	98 (0.2)
10 = (4,9)	89 (0.7)	96 (0.3)		
11 = (5,7)	32 (0.6)	48 (0.2)	67 (0.2)	
12 = (5,10)	63 (0.5)	99 (0.5)		
13 = (6,3)	66 (0.8)	85 (0.1)	98 (0.1)	
14 = (6,5)	6 (0.4)	15 (0.3)	39 (0.2)	58 (0.1)
15 = (6,7)	2 (0.6)	48 (0.4)		
16 = (7,8)	16 (0.3)	18 (0.3)	40 (0.2)	52 (0.2)
17 = (7,10)	16 (0.5)	34 (0.4)	71 (0.1)	
18 = (8,4)	90 (0.5)	96 (0.5)		
19 = (8,9)	17 (0.4)	49 (0.4)	53 (0.1)	65 (0.1)
20 = (9,7)	3 (0.5)	30 (0.4)	50 (0.1)	
21 = (9,10)	6 (0.5)	12 (0.3)	54 (0.1)	66 (0.1)

Table 1.3: Arc Capacity Distributions for Example 1.3.

Arc	Arc Capacity (Probability)			
1 = (1,2)	10 (0.3)	15 (0.7)		
2 = (1,3)	7 (0.2)	10 (0.2)	15 (0.6)	
3 = (1,4)	8 (0.5)	13 (0.5)		
4 = (2,5)	6 (0.2)	10 (0.3)	17 (0.5)	
5 = (2,6)	4 (0.1)	7 (0.2)	8 (0.3)	10 (0.4)
6 = (3,2)	7 (0.1)	9 (0.2)	15 (0.7)	
7 = (3,7)	5 (0.1)	8 (0.1)	10 (0.3)	16 (0.5)
8 = (3,8)	8 (0.1)	12 (0.9)		
9 = (4,3)	9 (0.1)	13 (0.2)	18 (0.7)	
10 = (4,9)	4 (0.2)	7 (0.2)	11 (0.2)	113 (0.4)
11 = (5,7)	8 (0.5)	14 (0.5)		
12 = (5,10)	6 (0.1)	15 (0.2)	17 (0.7)	
13 = (6,3)	4 (0.1)	7 (0.2)	12 (0.2)	20 (0.5)
14 = (6,5)	5 (0.3)	12 (0.7)		
15 = (6,7)	5 (0.2)	7 (0.3)	8 (0.5)	
16 = (7,8)	5 (1.0)			
17 = (7,10)	6 (0.2)	14 (0.8)		
18 = (8,4)	3 (0.1)	5 (0.1)	9 (0.2)	13 (0.6)
19 = (8,9)	8 (0.2)	10 (0.2)	14 (0.6)	
20 = (9,7)	4 (0.1)	7 (0.2)	9 (0.3)	12 (0.4)
21 = (9,10)	5 (0.3)	12 (0.7)		

minimum spanning trees using the probabilistic structure just described. (Local area networks, for example, often use a tree architecture.) Criticality indices will identify the links that are most likely to be economical.

Example 1.5 A given number of trucks are available to transport goods from city A to city B each day. Sometimes (according to historical probabilities which are known), one or more trucks are unavailable. Thus the capacity of the A-B transportation link depends on the number of available trucks, clearly a discrete random variable with finite state space. Investigating minimum cost flows in a network made up of many such shipping links will involve discrete random capacities.

Example 1.6 Often an excellent model for randomly-occurring phenomena is the Poisson process. Suppose that the occurrence of some delaying phenomenon along the length of a transportation link can be so modeled. Then the time it takes to traverse that link might depend on the number of such phenomena that are encountered while traversing the link and the time for traversing that link is clearly a discrete random variable.

The cardinality of the state space is the biggest obstacle in the evaluation of probabilities related to the operation of networks with randomly weighted components. For instance, the network in Table 1.1 has $3^{21} \approx 10$ billion states and the evaluation of the distribution of the weight of a MST by a complete enumeration of the state space requires 10 billion executions of a MST algorithm. If each arc has four possible settings, $4^{21} \approx 4$ quadrillion such executions are needed. The state space for the network in Example 1.3 has nearly 10^{19} states. It is this exponential explosion of work that has motivated the development of techniques which seek to evaluate only a fraction of these possible settings without sacrificing the quality of the resulting information.

In 1992, Ball *et. al.* surveyed state of the art methods for evaluating a broad assortment of network reliability and related problems [7]. In that paper they proposed the problems in Example 1.1 and in Example 1.3 as multistate performability

problems. After describing the stochastic maximum flow, shortest path and PERT problems, whose solutions they treated in some detail, the authors pointed out that "more elaborate models could include both cost and capacity information (random or deterministic) and include a wide variety of measures of system operation," with some examples being the stochastic minimum cost flow and minimum spanning tree problems. They found no satisfactory solution for these problems and suggested classes of algorithms to be applied to such unsolved problems. They stated that exact solution methods in network reliability typically involve convolutions or max(min) operations (arbitrary series of which are known to be #P-hard, even for variables which have only two possible values), or involve transformations into problems with two-state variables (a process that typically replaces each variable having q possible values by q two-state variables). Factoring, in which a state space is iteratively partitioned based on the state of a single variable, was also given as an important solution methodology. A variant of factoring was used in a paper by Doulliez and Jamouille [15] that solved problems related to maximum flows in networks having random discrete arc capacities. In that approach, the state space of arc capacities was iteratively decomposed based on *sets* of state points into known and unknown regions until the probabilistic behavior of all points in the state space was known. The survey by Ball *et. al.* concluded that all these available methods were necessarily inefficient since "essentially all reliability problems of interest are #P-complete." Thus it is seen that our proposed problems are not easy ones.

Some special cases of the MST problem have been treated in some detail. Gilbert [18] considered the problem of building MSTs to connect points randomly placed in the two-dimensional unit circle according to a Poisson process. Frieze [17] examined the case of the complete graph on n nodes whose arc costs are given as independent, identically distributed random variables. Assuming that the common distribution function satisfied certain smoothness requirements, he derived limiting behavior (as $n \rightarrow \infty$) of the expected MST cost. Steele [31] and Bertsimas [10] strengthened this result, known as the $\zeta(3)$ limit. Bertsimas and van Rysin [11] developed asymptotic

results for MSTs built to connect n points uniformly and independently distributed in the (multi-dimensional) unit ball. Kulkarni [23] used a clever Markov process approach to solve the special case of investigating minimum spanning trees when arc weights are independent exponential random variables. Bailey [5] extended these results to the general case of minimization on matroids with exponentially distributed element weights. Jain and Mamer [22] studied the problem of finding the mean and distribution of MST cost in a network whose arc costs are independent random variables. They developed an upper bound that proved to be better than the naive bound obtained by solving the deterministic MST with expected arc costs. Procedures for approximating the distribution function, which involved convolutions and Laplace-Stieltjes transforms, were demonstrated for exponentially distributed arc costs.

Thus, satisfactory solutions to the MST problems of Examples 1.1 and 1.2 do not appear in the literature. Specifically, no results deal with nonidentical discrete arc cost random variables or offer procedures applicable here for exactly determining the distribution of the MST cost. Furthermore, results relating to criticality indices as defined here in a MST context were examined only in [23].

Nor is the problem posed in Example 1.3 addressed in its full generality by existing solutions. Some results do exist for a few special cases, as we now discuss. The literature here is dominated by papers dealing with various forms of the stochastic maximum flow problem. The case in which only arc capacity is random has been treated by Doulliez and Jamoulle [15], Grimmett and Welsh [19], Fishman and Shaw [16], Alexopoulos and Fishman [3] [4], Alexopoulos [1], Nagamochi and Ibaraki [27] and others. Chen and Zhou [13] considered methods for system enhancement when only demand is random. Prékopa and Boros [30] evaluated the probability that the system of linear inequalities of Gale and Hoffman (which provide necessary and sufficient conditions for feasibility of demand) are satisfied. This work assumed random capacities and demand. Corea and Kulkarni [14] used first passage times in a particular Markov process to investigate minimum cost transshipment problems in

networks with independent exponential arc costs and no capacity limitations. Finally, the stochastic shortest path problem, where arc costs are random, was treated by Alexopoulos [2].

Thus none of the subject problems are satisfactorily treated in the literature, yet there is a need to produce the kind of information described above. Given the prospects of dramatically enlarging the state space with a transformation or of performing exponential numbers of convolutions of multistate variables, we are motivated by the factoring approach of Doulliez and Jamoulle. Similar approaches have been used to solve some other problems involving optimization of networks with stochastic components. See [3], [4] for maximum flow problems and [2] for shortest path problems. Furthermore, extensions have been made in the use of information obtained during a state space decomposition as input to highly efficient Monte Carlo sampling routines [1]. This study will describe a generalized version of this decomposition approach, and then use its expanded capabilities to solve probabilistic problems related to minimum spanning trees and minimum cost flows. These solutions have the attractive properties that exact computations typically require small numbers of iterations relative to the cardinality of the state space, and that the algorithms produce upper and lower bounds for the measures of interest that improve after each iteration.

Table 1.4 previews a few results from Chapter 3 for the MST model in Table 1.1 (FORTRAN 77 programs run on a SUN SPARCstation IPC) and Table 1.5 has representative results for minimum cost flow problems (FORTRAN 77 code executed on a SUN SPARCstation 2). Note that the number of iterations is only a diminutive fraction of the number of system states. The minimum spanning tree results extend naturally to other matroid problems, and the minimum cost flow results demonstrate the applicability of the proposed methods to a variety of stochastic network problems (e.g., maximum flow problems, shortest path problems, assignment problems, matching problems, etc.) that can be reduced to minimum cost flow problems.

Table 1.4: Sample of decomposition results for Examples 1.1 and 1.2.

Information computed exactly	# iterations	CPU time (sec)
$P\{\text{MST cost} \leq 60\} = 0.8267496$	1421	1.30
$P\{\text{MST cost} \leq 90\} = 0.9999951$	127,683	95.18
Entire distribution of MST cost	634,405	445.91
$P\{\text{arc 1 is on a MST}\} = 0.9393247$	12	0.71
$P\{\text{arc 5 is on a MST}\} = 0.8227428$	200	1.18
$P\{\text{arc 14 is on a MST}\} = 0.0099071$	169	1.00

Table 1.5: Sample of decomposition results for Example 1.3.

Information computed exactly	# iterations	CPU time (sec)
$P\{\text{MCF cost} \leq 1900\} = 0.30378595$	78,488	310.25
$P\{\text{MCF cost} \leq 3500\} = 0.99999962$	132,944	443.50

In Chapter 2 we begin by describing the General State Space Decomposition algorithm and point out its utility in conducting probabilistic analysis, not necessarily in the area of network optimization. In Chapter 3 we propose a number of different algorithms for stochastic minimum spanning tree problems and discuss extensions to general matroid settings. Chapter 4 contains algorithms for stochastic minimum cost flow problems. We conclude in Chapter 5 with topics for future research. Numerical examples are given throughout this study.

CHAPTER 2

General State Space Decomposition Algorithm

2.1 Introduction

In this chapter we describe methods for evaluating probabilities related to systems of independent discrete random variables. Faced with such a task, one must devise ways to combat the associated computational intractability mentioned in Chapter 1. Solution methods involving state enumeration are clearly out of the question since the state spaces of these systems grow exponentially with the number of variables.

One possible approach is Monte Carlo simulation. In crude Monte Carlo, state points are sampled according to the probability distributions of the input random variables. The proportion of sampled points having a desired property is an estimate of the probability that the modeled system has that property. Unfortunately, in order to have an estimate with acceptably small variance, an astronomical number of samples must be drawn.

Much work in reliability analysis (a special case of the present setting in which system components are given by binary absence/presence variables) has used a factoring (or, pivotal decomposition) approach, in which the state space is iteratively conditioned and partitioned based on the state of a single component [8]. Basically, factoring iteratively makes use of the following fact (where the event O corresponds

to the system operating and the event O_i corresponds to component i operating):

$$P\{O\} = P\{O|O_i\}P\{O_i\} + P\{O|\bar{O}_i\}P\{\bar{O}_i\}.$$

These procedures have had good success.

When the random variables describing the system are allowed to take on arbitrary values, though, performance can suffer. This situation can grow still worse if we allow the variables to have arbitrary (finite) numbers of possible values (as we shall do in the present general setting). Though factoring in this setting has been used effectively in some cases ([21], for example), in general it is only effective if the system being studied has a special structure that can be exploited.

In 1972, in a paper dealing with problems related to stochastic maximum flows, Doulliez and Jamoulle [15] used an adaptation of the factoring approach to classify and decompose the state space based on *sets* of state points. (The connection between factoring and our algorithm will be explained in Section 2.5.) Since then, similar approaches have been used in other stochastic network settings [1],[2],[3],[4]. The purpose of this chapter is to describe what we shall refer to as the General State Space Decomposition (GSD) algorithm, with the following goals in mind:

- 1) To define terms necessary to build a general framework within which we may discuss these methods;
- 2) To describe a general input problem broad enough to accomodate all known applications of the Doulliez-Jamoulle decomposition approach as well as new classes of problems both within and without the realm of network analysis;
- 3) To describe GSD, which should not only expand existing methods to include the broader problem class, but should also enhance them in ways that can lead to more efficiently computed bounds on the distributions of interest; and
- 4) To derive theoretical results comparing GSD to alternative decomposition strategies.

We shall see that GSD has a number of important advantages over alternative methods. (1) Upper and lower bounds on the probability of interest are available after each iteration, and these bounds are progressively tighter. (2) The information available in each iteration can be used for constructing efficient Monte Carlo procedures based on importance and stratified sampling. (3) Sensitivity analysis can be accomplished very easily when input probabilities change, with no additional decomposition effort. (4) GSD is maximally efficient (with respect to minimizing the number of partitions formed per iteration) among a large class of methods.

We begin in Section 2.2 with some definitions and mathematical preliminaries leading up to a general problem statement in Section 2.3. Section 2.4 contains the GSD algorithm. Section 2.5 establishes theoretical results concerning the partitioning of the state space. Section 2.6 discusses the use of GSD to produce bounds, and Section 2.7 shows how these bounds can be used as inputs to Monte Carlo estimation routines. Concluding remarks are given in Section 2.8.

2.2 Definitions and Preliminaries

Consider a vector $\mathbf{X} = (X_1, \dots, X_a)$ of discrete random variables with finite state spaces. Each component X_j takes on finite values $w_j(1) < \dots < w_j(n_j)$ (which we refer to as *weights*) with respective probabilities $p_j(1), \dots, p_j(n_j)$. An element of Ω , the state space of \mathbf{X} , is of the form $x = (w_1(l_1), \dots, w_a(l_a))$, where $l_j \in \{1, \dots, n_j\}$ for all j . In the interest of readability, we shall often refer to the state point x as simply $l = (l_1, \dots, l_a)$, letting the *index* l_j stand for $w_j(l_j)$. Let $\mathcal{A} = \{1, \dots, a\}$.

Unless otherwise stated, we assume that the random variables X_j are mutually independent, so that the probability of the state point x is calculated easily as

$$\begin{aligned} P(x) &= P\{X_1 = w_1(l_1), \dots, X_a = w_a(l_a)\} \\ &= \prod_{j=1}^a p_j(l_j). \end{aligned} \tag{2.1}$$

Often we will be dealing not with individual state points, but with special types

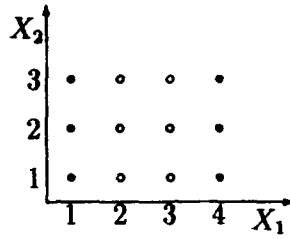


Figure 2.1: Discrete interval given in Example 2.1.

of sets which we call *discrete intervals*.

Definition 2.1 A subset of Ω is a *discrete (multi-dimensional) interval* with endpoints α and β ($\alpha \leq \beta$) if it consists precisely of all state points $l = (l_1, \dots, l_a)$ satisfying $l_j \in \{\alpha_j, \dots, \beta_j\}$ for all j . We denote this interval by $[\alpha, \beta]$.

Example 2.1 Let $a = 2$, $n_1 = 4$, and $n_2 = 3$. Then there are $4 \times 3 = 12$ points in Ω , as shown in Figure 2.1. The state points shown as open circles constitute the interval $[\alpha, \beta]$ with endpoints $\alpha = (2, 1)$ and $\beta = (3, 3)$.

The probability of an interval can be easily calculated, again due to independence of the random variables, as

$$P\{\mathbf{X} \in [\alpha, \beta]\} = P\{[\alpha, \beta]\} = \prod_{j=1}^a \sum_{i=\alpha_j}^{\beta_j} p_j(i). \quad (2.2)$$

This ease of calculation, and the fact that intervals can be stored using just the endpoints α and β , make intervals very attractive. GSD deals exclusively with intervals.

Suppose that the vector \mathbf{X} represents the states of the components of a system which operates when a set of structural constraints $S(\mathbf{X})$ is satisfied. We further assume that the validity of these constraints (i.e., whether or not these constraints are satisfied) can be checked for any realization of \mathbf{X} .

The following definitions set the stage for the formal description of GSD.

Definition 2.2 Any point l in the state space Ω which satisfies all constraints in the set $S(\cdot)$ is a *feasible point*. If l does not satisfy $S(\cdot)$, then l is an *infeasible point*.

Definition 2.3 A set whose states have been classified as feasible (infeasible) is called *feasible (infeasible)*. Sets with unclassified states will be called *undetermined*.

Definition 2.4 The maximal feasible (infeasible) subset of Ω is called the *feasible (infeasible) region* and is denoted Ω^F (Ω^I). Clearly, Ω is partitioned as $\Omega = \Omega^F \cup \Omega^I$.

The purpose of our algorithm will be to compute the probability that the system operates (i.e., that the system is in a feasible state), denoted $P^F \triangleq P\{\Omega^F\}$. We shall refer to this measure as the *system reliability*.

Thus far, we have not restricted in any way the relationship between $S(\mathbf{X})$ and the individual random variables X_j . We must now do so by requiring a sort of monotonicity of the satisfaction of $S(\mathbf{X})$ with respect to each X_j . Formally, we require that it be possible to unambiguously classify each X_j as either *upper feasible* or *lower feasible*, as defined below.

Definition 2.5 X_j will be called *lower feasible* if for every realization of $X_i, i \neq j$, all levels of X_j corresponding to feasible state points are less than all levels of X_j corresponding to infeasible state points.

Definition 2.6 X_j will be called *upper feasible* if for all realizations of $X_i, i \neq j$, all levels of X_j corresponding to feasible state points are greater than all levels of X_j corresponding to infeasible state points.

This classification induces a partition of the components of \mathbf{X} . Henceforth, we assume that the random variables are numbered so that lower feasible variables are numbered as X_1, \dots, X_m while the upper feasible variables are numbered as X_{m+1}, \dots, X_n . We shall often emphasize this distinction by writing a state l in partitioned form as

$$l = (l_1, \dots, l_m | l_{m+1}, \dots, l_a).$$

Imposing the above restriction on \mathbf{X} and $S(\mathbf{X})$ was necessary to establish the following useful result.

Proposition 2.1 *Let \mathbf{X} be a vector of a independent discrete random variables with variables X_1 through X_m being lower feasible and variables X_{m+1} through X_a being upper feasible. Then for all $\bar{l} \in \Omega$:*

(i) *If \bar{l} is feasible, then the set*

$$F = \{l \in \Omega : 1 \leq l_i \leq \bar{l}_i \text{ for } i = 1, \dots, m \\ \bar{l}_i \leq l_i \leq n_i \text{ for } i = m+1, \dots, a\}$$

is feasible.

(ii) *If \bar{l} is infeasible, then the set*

$$I = \{l \in \Omega : \bar{l}_i \leq l_i \leq n_i \text{ for } i = 1, \dots, m \\ 1 \leq l_i \leq \bar{l}_i \text{ for } i = m+1, \dots, a\}$$

is infeasible.

Proposition 2.1 is valuable because it allows us to classify entire intervals in Ω as feasible or infeasible based solely on the feasibility of a single point. It also polarizes each interval so that it has a "feasible end" and an "infeasible end."

Definition 2.7 Consider an interval $[\alpha, \beta]$ in Ω . The point

$$(\alpha|\beta) = (\alpha_1, \dots, \alpha_m | \beta_{m+1}, \dots, \beta_a)$$

will be called the *extreme feasible candidate* at the point

$$(\beta|\alpha) = (\beta_1, \dots, \beta_m | \alpha_{m+1}, \dots, \alpha_a)$$

will be called the *extreme infeasible candidate*.

Note that this definition does not imply that the extreme feasible candidate is a feasible point or that the extreme infeasible candidate is an infeasible point. If the extreme feasible candidate is found to be feasible, we may refer to it as the *feasible extreme*. Likewise for the infeasible extreme. Indeed, Proposition 2.1 implies that if $(\alpha|\beta)$ is infeasible then all of $[\alpha, \beta]$ is infeasible. Similarly, if $(\beta|\alpha)$ is feasible then so is all of $[\alpha, \beta]$. Thus Definition 2.7, in conjunction with Proposition 2.1, hints at a key idea of GSD: First try to classify an interval as feasible or infeasible by evaluating its extremes. If this fails, start at the feasible (or infeasible) extreme and push in as far as possible while remaining feasible (infeasible), yielding a large feasible (infeasible) interval. Following the general problem statement in Section 2.3, we shall describe in Section 2.4 the manner in which this idea is put to use.

Before getting into the technical details of GSD, a comment concerning system modeling is in order. Without significant modification GSD has the capability for limited representation of simple dependencies among variables. Specifically, some of the variables X_1, \dots, X_n can represent *vectors* of dependent variables. This technique can result in more realistic modeling (as opposed to treating these attributes as independent) and at the same time reduces the cardinality of the state space. (For example, replacing 3 variables, each with 4 levels, by a triple having 4 levels reduces the cardinality of Ω by a factor of $4^3 = 64$.) Of course, the realizations of X_j must be chosen and ordered so that X_j can be classified as upper or lower feasible. For example, in Chapter 4 we shall use $X_j = (\text{cost of arc } j, \text{ capacity of arc } j)$ in the stochastic minimum cost flow setting. By themselves, *cost* is lower feasible and *capacity* is upper feasible. As an ordered pair, then, high capacity must be paired with low cost so that they may be jointly classified. Fortunately, it is reasonable in many applications that units of scarce capacity are expensive or cause extended delays.

2.3 General Problem Statement

Recapitulating, we have a stochastic system represented by a vector $\mathbf{X} = (X_1, \dots, X_m | X_{m+1}, \dots, X_a)$ of a mutually independent discrete random variables with finite support. We wish to calculate the system reliability, defined as the probability that the constraints $S(\cdot)$ are satisfied (i.e., the probability of the feasible region Ω^F). We assume that the variables X_1 through X_m are lower feasible while the variables X_{m+1} through X_a are upper feasible. This causes Ω to be ordered in the sense of Proposition 2.1. We proceed to describe the General State Space Decomposition algorithm.

2.4 The GSD Algorithm

We are now prepared to discuss a typical iteration of GSD, in which an undetermined set is partitioned into (non-overlapping) feasible, infeasible and undetermined intervals. The probabilities of the feasible intervals are accumulated as a lower bound for P^F , the infeasible intervals are discarded, and the undetermined intervals are filed in a list as input to subsequent iterations. Following the description, in Section 2.5 we present theoretical results showing the performance of this algorithm to be optimal in terms of producing fewest new undetermined intervals per iteration.

There are two basic approaches. *Decomposition from the feasible extreme point* (or, *feasible decomposition*) begins at the feasible extreme and attempts to push in as far as possible, producing one large feasible interval. *Decomposition from the extreme infeasible point* (or, *infeasible decomposition*), on the other hand, drives in from the infeasible extreme to remove one large infeasible interval. We shall describe feasible decomposition in detail: infeasible decomposition proceeds symmetrically.

We begin with an undetermined set $U = [\alpha, \beta]$. As a preliminary screening, evaluations are made at both extremes of the set. If $S(\cdot)$ is not satisfied at $(\alpha|\beta)$ (that is, if the extreme feasible candidate is an infeasible point), then by Proposition 2.1 the entire set is infeasible, it is discarded and the iteration is ended. Otherwise,

the state $(\alpha|\beta)$ is feasible. If $(\beta|\alpha)$ also satisfies $S(\cdot)$, then by Proposition 2.1 U is feasible, the probability of U is added to the accumulated lower bound for P^F , and again the iteration is ended. If not, then U clearly contains both feasible and infeasible points and we proceed.

The major step of feasible decomposition is the identification of a feasible set. The exact mechanics of this step depend on the particular problem being solved. We seek a feasible point \bar{l} which is easily derived and which will yield as large a feasible set as possible. We begin with our only known feasible point, $(\alpha|\beta)$. We then increase the weights of the lower feasible components and decrease those of the upper feasible components as much as possible while remaining feasible.

In practice, we generally stop short of "as much as possible" in the interest of efficiency. Often a fairly deep cut-off state can be derived with reasonable effort, and to push a moderate amount further would incur significant cost. Additionally, in many cases it may be difficult even to determine the extent to which further pushing is possible. Fortunately this is not necessary, for Proposition 2.1 guarantees that *any* feasible point yields a feasible interval. Just how much can be removed at a time will likely depend on the problem being solved.

After completing this step, we obtain a feasible interval

$$F = \{l \in U : \alpha_j \leq l_j \leq \bar{l}_j \text{ for } j = 1, \dots, m \\ \bar{l}_j \leq l_j \leq \beta_j \text{ for } j = m+1, \dots, a\} \quad (2.3)$$

and add the probability of F

$$P\{F\} = \left[\prod_{j=1}^m \sum_{i=\alpha_j}^{\bar{l}_j} p_j(i) \right] \left[\prod_{j=m+1}^a \sum_{i=\bar{l}_j}^{\beta_j} p_j(i) \right] \quad (2.4)$$

to the lower bound for P^F .

We can stop after determining F , breaking up the remainder of U into up to a undetermined intervals as we indicate below. In some applications of GSD, we shall do precisely this. However, since we wish to account for as much undetermined probability as possible in each iteration, it makes sense to try to remove infeasible sets as well. This cannot be done naively. As we shall soon see, blindly removing even one infeasible set in addition to F can increase the number of new undetermined sets by as much as a . The following technique removes up to a infeasible sets in addition to F and *cannot* increase the number of undetermined sets already needed to partition $U \setminus F$. (In fact, the number can be reduced since some of the sets in the partition of $U \setminus F$ can be found to be infeasible.)

In order to identify infeasible sets, we turn again to Proposition 2.1. This step is carried out individually and independently for each of the a components. For component j , this step consists of starting at the extreme feasible point $(\alpha|\beta)$ and pushing *only* this component (increasing its weight if it is lower feasible, decreasing its weight if it is upper feasible) as far as possible while remaining feasible. Here it is essential that we push as far as possible.

Definition 2.8 We define the *limiting feasible index* for component j by

$$\hat{l}_j = \max\{\gamma : \alpha_j \leq \gamma \leq \beta_j, (\alpha_1, \dots, \alpha_{j-1}, \gamma, \alpha_{j+1}, \dots, \alpha_m | \beta) \text{ is feasible}\}$$

if X_j is lower feasible, and by

$$\hat{l}_j = \min\{\gamma : \alpha_j \leq \gamma \leq \beta_j, (\alpha | \beta_{m+1}, \dots, \beta_{j-1}, \gamma, \beta_{j+1}, \dots, \beta_a) \text{ is feasible}\}$$

if X_j is upper feasible.

Note that for lower feasible variables j we have $\hat{l}_j \geq \bar{l}_j$ so that if $\bar{l}_j = \beta_j$ we can set $\hat{l}_j = \beta_j$ with no additional effort. Similarly, for upper feasible variables j we have $\hat{l}_j \leq \bar{l}_j$, allowing us to set $\hat{l}_j = \alpha_j$ if $\bar{l}_j = \alpha_j$.

If we are able to push \hat{l}_j all the way (to β_j if X_j is lower feasible, to α_j if X_j is upper feasible), no infeasible set is produced. Otherwise, we know that pushing

the weight of component j one more level while leaving all other variables at $(\alpha|\beta)$ yields an infeasible point, and hence an infeasible set. If X_j is lower feasible (that is, if $j \leq m$), the infeasible set is

$$\begin{aligned} \hat{I}_j = \{l \in U : & \hat{l}_j + 1 \leq l_j \leq \beta_j \\ & \alpha_i \leq l_i \leq \beta_i \quad \text{for } i \neq j\}, \end{aligned} \quad (2.5)$$

while if X_j is upper feasible ($j \geq m+1$) this set is

$$\begin{aligned} \hat{I}_j = \{l \in U : & \alpha_j \leq l_j \leq \hat{l}_j - 1 \\ & \alpha_i \leq l_i \leq \beta_i \quad \text{for } i \neq j\}. \end{aligned} \quad (2.6)$$

Note that the sets \hat{I}_j are not disjoint (indeed, all contain the point $(\beta|\alpha)$). In most cases this is not a problem since the infeasible sets are discarded *en masse*. In some applications, though, infeasible sets are retained for further analysis, as we discuss later. In those cases it is necessary to partition the infeasible sets. Recall that the union of non-disjoint sets $\hat{A}_1, \dots, \hat{A}_a$ can be partitioned into the (disjoint) sets

$$\begin{aligned} A_1 &= \hat{A}_1 \\ A_2 &= \hat{A}_2 \setminus \hat{A}_1 \\ A_3 &= \hat{A}_3 \setminus (\hat{A}_1 \cup \hat{A}_2) \\ &\vdots \\ A_a &= \hat{A}_a \setminus (\hat{A}_1 \cup \dots \cup \hat{A}_{a-1}). \end{aligned}$$

Applying this principle to the present setting, we obtain new infeasible sets I_j as follows. If X_j is lower feasible, then

$$\begin{aligned} I_j = \{l \in U : & \alpha_i \leq l_i \leq \hat{l}_i \quad \text{for } i < j \\ & \hat{l}_j + 1 \leq l_j \leq \beta_j \\ & \alpha_i \leq l_i \leq \beta_i \quad \text{for } i > j\}, \end{aligned} \quad (2.7)$$

whereas if X_j is upper feasible, then

$$I_j = \{l \in U : \alpha_i \leq l_i \leq \hat{l}_i \quad \text{for } i \leq m$$

$$\begin{aligned}
\hat{l}_i \leq l_i \leq \beta_i & \quad \text{for } m < i < j \\
\alpha_j \leq l_j \leq \hat{l}_j - 1 & \\
\alpha_i \leq l_i \leq \beta_i & \quad \text{for } i > j \}.
\end{aligned} \tag{2.8}$$

Notice that the resulting infeasible sets are intervals.

We have thus found one feasible set (by finding the feasible point \bar{l}) and up to a infeasible sets (by finding the limiting feasible index \hat{l}_j for each j). All that remains is to account for the portion of U that remains undetermined. We wish to represent $U \setminus (F \cup (\cup_{j=1}^a I_j))$ as a disjoint union of intervals. We account for the points "between" F and I_j for each j by defining the set

$$\begin{aligned}
\tilde{U}_j = \{l \in U : & \bar{l}_j + 1 \leq l_j \leq \hat{l}_j \\
& \alpha_i \leq l_i \leq \hat{l}_i \quad \text{for } i \neq j, i \leq m \\
& \hat{l}_i \leq l_i \leq \beta_i \quad \text{for } i \neq j, i > m\} , \text{ if } j \leq m
\end{aligned} \tag{2.9}$$

$$\begin{aligned}
= \{l \in U : & \hat{l}_j \leq l_j \leq \bar{l}_j - 1 \\
& \alpha_i \leq l_i \leq \hat{l}_i \quad \text{for } i \neq j, i \leq m \\
& \hat{l}_i \leq l_i \leq \beta_i \quad \text{for } i \neq j, i > m\} , \text{ if } j > m.
\end{aligned}$$

To verify this, note that putting component j beyond \bar{l}_j is sufficient to yield points outside of F independent of the levels of the other components. Also, no component is permitted to be at a level beyond the index \hat{l} since this would yield points already accounted for by $\cup_{j=1}^a I_j$.

Unfortunately, the undetermined sets \tilde{U}_j overlap and their union must be partitioned. This is done by defining the intervals

$$\begin{aligned}
U_j = \{l \in U : & \alpha_i \leq l_i \leq \bar{l}_i \quad \text{for } i < j \\
& \bar{l}_j + 1 \leq l_j \leq \hat{l}_j \\
& \alpha_i \leq l_i \leq \hat{l}_i \quad \text{for } j < i \leq m
\end{aligned}$$

$$\hat{l}_i \leq l_i \leq \beta_i \quad \text{for } i > m \}, \text{ if } j \leq m \quad (2.10)$$

$$\begin{aligned} = \{l \in U : & \quad \alpha_i \leq l_i \leq \bar{l}_i \quad \text{for } i \leq m \\ & \quad \bar{l}_i \leq l_i \leq \beta_i \quad \text{for } m < i < j \\ & \quad \hat{l}_j \leq l_j \leq \bar{l}_j - 1 \\ & \quad \hat{l}_i \leq l_i \leq \beta_i \quad \text{for } i > j \}, \text{ if } j > m. \end{aligned}$$

It is now easy to show that $U = F \cup \{U_{j=1}^a I_j\} \cup \{U_{j=1}^a U_j\}$, thus completing the partition of U . If we chose not to remove infeasible sets, so that limiting indices were not found, we replace \hat{l}_i in this description by β_i if $i \leq m$, or by α_i if $i > m$. We calculate $P\{F\}$ using (2.4) and add the result to P_l^F , the accumulated lower bound for P^F . The undetermined sets are filed in their list, and the infeasible sets are discarded (or retained for further analysis). The iteration is over. An undetermined interval is then selected from the list and the next iteration begins.

To summarize, procedure FEASIBLE(U) describes the decomposition of a single undetermined set $U = [\alpha, \beta]$. The list \mathcal{U} contains all remaining undetermined intervals.

Procedure FEASIBLE(U)

Step 1 Start with the state $(\alpha|\beta)$ and, while $S(\cdot)$ remains satisfied, determine a feasible state \bar{l} by pushing up from α_j for $j \leq m$ and down from β_j for $j > m$.

Step 2 Determine limiting feasible indices \hat{l}_j as in Definition 2.8. (If infeasible sets are not to be removed, set $\hat{l} = (\beta|\alpha)$.)

Step 3 Set $P_l^F = P_l^F + P\{F\}$, where $P\{F\}$ is calculated as in (2.4).

Step 4 For $j = 1, \dots, a$: If $\hat{l}_j \neq l_j$, file U_j in \mathcal{U} , where U_j is defined by (2.10).

As indicated previously, decomposition of a given interval can proceed from either the extreme feasible point or the extreme infeasible point. Infeasible decomposition

is completely symmetric to feasible decomposition. We find an infeasible point \bar{l} by starting from $(\beta|\alpha)$, and define the infeasible interval

$$I = \{l \in U : \bar{l}_j \leq l_j \leq \beta_j \text{ for } j = 1, \dots, m \quad (2.11)$$

$$\alpha_j \leq l_j \leq \bar{l}_j \text{ for } j = m+1, \dots, a\}.$$

If we also desire to remove feasible sets, we continue as follows.

Definition 2.9 We define the *limiting infeasible index for component j* by

$$\hat{l}_j = \min\{\gamma : \alpha_j \leq \gamma \leq \beta_j, (\beta_1, \dots, \beta_{j-1}, \gamma, \beta_{j+1}, \dots, \beta_m | \alpha) \text{ is infeasible}\},$$

if X_j is lower feasible, and by

$$\hat{l}_j = \max\{\gamma : \alpha_j \leq \gamma \leq (\beta | \alpha_{m+1}, \dots, \alpha_{j-1}, \gamma, \alpha_{j+1}, \dots, \alpha_a) \text{ is infeasible}\}$$

if X_j is upper feasible.

Note that in infeasible decomposition we have $\hat{l}_j \leq \bar{l}_j$ for $j \leq m$ and $\hat{l}_j \geq \bar{l}_j$ for $j > m$.

The indices \hat{l}_j define up to a disjoint feasible intervals F_j . If X_j is lower feasible (i.e., if $j \leq m$), then

$$F_j = \{l \in U : \hat{l}_i \leq l_i \leq \beta_i \text{ for } i < j$$

$$\alpha_j \leq l_j \leq \hat{l}_j - 1 \quad (2.12)$$

$$\alpha_i \leq l_i \leq \beta_i \text{ for } i > j\},$$

whereas if X_j is upper feasible ($j > m$), then

$$F_j = \{l \in U : \hat{l}_i \leq l_i \leq \beta_i \text{ for } i \leq m$$

$$\alpha_i \leq l_i \leq \hat{l}_i \text{ for } m < i < j$$

$$\hat{l}_j + 1 \leq l_j \leq \beta_j \quad (2.13)$$

$$\alpha_i \leq l_i \leq \beta_i \text{ for } i > j\}.$$

Note that $F_j = \emptyset$ if $j \leq m$ and $\hat{l}_j = \alpha_j$ or if $j > m$ and $\hat{l}_j = \beta_j$.

What remains is partitioned into disjoint undetermined intervals U_j given by

$$\begin{aligned}
 U_j = \{l \in U : & \quad \bar{l}_i \leq l_i \leq \beta_i \quad \text{for } i < j \\
 & \quad \hat{l}_j \leq l_j \leq \bar{l}_j - 1 \\
 & \quad \hat{l}_i \leq l_i \leq \beta_i \quad \text{for } j < i < m \\
 & \quad \alpha_i \leq l_i \leq \hat{l}_i \quad \text{for } i > m\}, \quad \text{if } j \leq m \\
 & \\
 = \{l \in U : & \quad \bar{l}_i \leq l_i \leq \beta_i \quad \text{for } i \leq m \\
 & \quad \alpha_i \leq l_i \leq \bar{l}_i \quad \text{for } m < i < j \\
 & \quad \bar{l}_j + 1 \leq l_j \leq \hat{l}_j \\
 & \quad \alpha_i \leq l_i \leq \hat{l}_i \quad \text{for } i > j\}, \quad \text{if } j > m.
 \end{aligned} \tag{2.14}$$

Note that, as before, U_j is empty if $\bar{l}_j = \hat{l}_j$. Otherwise, U_j will be maintained for future iterations. (Again, if limiting indices were not found, we replace \hat{l}_i by α_i if $i \leq m$ and by β_i if $i > m$ in the description of U_j .)

We summarize the infeasible decomposition below (again, pre-screening has ascertained that the interval being decomposed, $U = [\alpha, \beta]$, contains both feasible and infeasible points).

Procedure INFEASIBLE(U)

Step 1 Start with the state $(\beta|\alpha)$ and, while $S(\cdot)$ is not satisfied, determine an infeasible state \bar{l} by pushing down from β_j for $j \leq m$ and up from α_j for $j > m$.

Step 2 Determine limiting infeasible indices \hat{l}_j as in Definition 2.9. (If feasible sets are not to be removed, set $\hat{l} = (\alpha|\beta)$.)

Step 3 For $j = 1, \dots, a$: Form F_j as in (2.12) or (2.13). If $F_j \neq \emptyset$, set $P_l^F = P_l^F + P\{F_j\}$, where $P\{F_j\}$ is calculated as in (2.2).

Step 4 For $j = 1, \dots, a$: If $\bar{l}_j \neq \bar{u}_j$, file U_j in \mathcal{U} , where U_j is defined by (2.14).

Finally, we give the entire General State Space Decomposition algorithm.

Procedure GSD

Step 1 Start with $U = \Omega$. Set $\mathcal{U} = \emptyset$ and $P_l^F = 0$.

Step 2 Let α and β denote, respectively, the lower and upper end points of U . If $(\alpha|\beta)$ is not feasible, go to step 4. If $(\beta|\alpha)$ is feasible, set $P_l^F = P_l^F + P\{U\}$ and go to step 4.

Step 3 Perform either FEASIBLE(U) or INFEASIBLE(U).

Step 4 If $\mathcal{U} = \emptyset$, stop. Otherwise, remove a set U from \mathcal{U} and go to step 2.

The statement of step 3 is noticeably vague. How should we decide whether to decompose a given interval feasibly or infeasibly? We consider the following options:

Pure Option 1 All intervals are decomposed from the feasible extreme.

Pure Option 2 All intervals are decomposed from the infeasible extreme.

Mixed Option 1 Each interval is decomposed both ways. The decomposition for which $\sum_{j=1}^a P(U_j)$ is smaller is used.

Mixed Option 2 Each interval is decomposed both ways. The decomposition which yields fewer new undetermined intervals is used.

Mixed Option 3 Each interval is decomposed both ways. The decomposition that produces new undetermined intervals containing fewer state points is used.

The idea behind the mixed options is to choose the "best" decomposition in each iteration. In practice, they seem to perform better than either pure strategy in some cases. However, it also seems clear that trying to "optimize" with regard

to one of the above criteria in each step (filing less probability, filing fewer points, filing fewer sets) in no way guarantees the most efficient possible evaluation of P^F . Some or all of the five options will be compared in Chapter 3 for minimum spanning tree problems and in Chapter 4 for minimum cost network flow problems.

Another practical consideration deals with the handling of the list of undetermined sets. Again, a number of possibilities exist. Alexopoulos [1] suggested that LIFO queue discipline outperforms FIFO in keeping the list small, and that keeping a list sorted by set probability works well when the objective is the computation of tight bounds. We have had very good results keeping a sorted list initially (with the most probable undetermined interval on top), switching to LIFO when the probability of the most probable filed set is suitably small. All numerical results in this thesis use this form of list management. Sorting the list by probability is especially helpful in yielding tight bounds on P^F more quickly, as discussed in Section 2.6.

Finally, we have stated that feasible and infeasible intervals produced in each iteration are discarded at the end of that iteration. There may be situations in which it is useful to save one or both of these set types in files for future use. (Recall that each interval is stored using $2a$ integer memory locations since we need only the lower and upper end points.) For example, if the possible values $w_j(\cdot)$ are known for each X_j but the respective probabilities either are not known or are subject to change, we may retain the feasible sets (which will still represent a partitioning of the feasible region) and easily re-calculate P^F based on the new probabilities. This is very helpful in performing sensitivity analyses. At other times, we might wish to retain the infeasible sets for further analysis. For example, in Chapter 3 we discuss passing the infeasible sets from the evaluation of the probability that the weight of a MST does not exceed d as input to the routine that calculates criticality indices so as to compute the probability that an arc e is on a MST whose weight exceeds d .

2.5 Theoretical Results Concerning GSD

In this section, we digress to tie up some theoretical loose ends concerning the General State Space Decomposition algorithm. First, we demonstrate the connection between our algorithm and factoring. Without loss of generality, we consider decomposition from the feasible extreme. In each iteration we decompose an undetermined set U into a feasible set F , infeasible sets $\{I_j\}_{j \in \mathcal{A}}$ and new undetermined sets $\{U_j\}_{j \in \mathcal{A}}$. Let U^F denote the feasible portion of U . From a factoring standpoint, then, we have done the following.

$$\begin{aligned} P\{U^F\} &= P\{U^F \cap F\} + \sum_{j=1}^a P\{U^F \cap I_j\} + \sum_{j=1}^a P\{U^F \cap U_j\} \\ &= P\{F\} + 0 + \cdots + 0 + P\{U^F \cap U_1\} + \cdots + P\{U^F \cap U_a\} \\ &= P\{F\} + P\{U_1^F\} + \cdots + P\{U_a^F\}. \end{aligned}$$

$P\{F\}$ is added to the accumulated lower bound for P^F . The sets U_j are filed for future iterations, so that $P\{U_j^F\}$ can be calculated in precisely this way.

Secondly, we present remarks concerning the number of new undetermined intervals produced in each iteration of GSD. In particular, we shall investigate the consequences of removing various combinations of feasible and infeasible sets, concluding that the methods given in Section 2.4 cannot be improved upon in this regard. We shall look chiefly at what we have called decomposition from the feasible extreme. By symmetry, the results also apply to decomposition from the infeasible extreme. For convenience, and without loss of generality, we assume in this section that all variables are lower feasible, and that we seek to decompose a set $U = [\alpha, \beta]$ for which α is a feasible point and β is an infeasible point.

Our first result concerns the number of undetermined intervals needed to partition what remains of U after the removal of a single feasible set of the form described in Proposition 2.1. The symmetric result for infeasible sets is implied.

Proposition 2.2 *Let $\bar{l} \in U$ be a feasible point which differs from the point β in i positions ($0 \leq i \leq a$), and let F be the feasible set defined by \bar{l} using Proposition 2.1. Then exactly i intervals are required to partition $U \setminus F$.*

Proof: We have already seen that using (2.10) GSD produces one undetermined interval for each $j \in \mathcal{A}$ with $\bar{l}_j < \beta_j$. In the present setting, then, i intervals would be produced. Thus we need only show that it is not possible to partition $U \setminus F$ using fewer than i intervals. We shall do this by using mathematical induction on a , the number of components.

The result holds for $a = 2$, as we now demonstrate. If $i = 0$, then $\bar{l} = \beta$ and $U \setminus F = \emptyset$. If $i = 1$, then \bar{l} differs from β in one position, and hence $U \setminus F \neq \emptyset$, so that it is not possible to cover $U \setminus F$ with 0 sets. Finally, if $i = 2$ then clearly an "L-shaped" lattice remains, and it is thus impossible to cover $U \setminus F$ with 0 or 1 interval.

Now suppose the result holds for $a - 1$ variables (X_1, \dots, X_{a-1}) and consider the case for a variables. It will be useful to view Ω_a , the state space of (X_1, \dots, X_a) , as consisting of n_a ordered copies of Ω_{a-1} , the state space of (X_1, \dots, X_{a-1}) , indexed by $\{1, \dots, n_a\}$, and to view undetermined intervals U as being similarly made up of ordered lower-dimensional undetermined intervals.

The result clearly holds when $i = 0$ or $i = 1$ by precisely the same logic as was used above. Now suppose that $2 \leq i \leq a$. We prove that it is impossible to partition $U \setminus F$ using fewer than i intervals by contradiction. In light of our previous remark, U will be viewed as consisting of ordered $(a - 1)$ -dimensional undetermined intervals designated by $U_{\alpha_a}, \dots, U_{\beta_a}$. The set F resides in sets $U_{\alpha_a}, \dots, U_{\bar{l}_a}$, where it forms similar sets $F_{\alpha_a}, \dots, F_{\bar{l}_a}$ (where by *similar* we mean that each F_j is oriented with respect to the other points in U_j in the same way). We shall consider separately the cases $\bar{l}_a = \beta_a$ and $\bar{l}_a < \beta_a$.

If $\bar{l}_a = \beta_a$, then clearly $i \leq a - 1$ (\bar{l} agrees with β in at least one position). In U_{β_a} , the point \bar{l} differs from the point β in i positions. Any partition of $U \setminus F$ consisting of fewer than i intervals clearly implies a partition of $U_{\beta_a} \setminus F_{\beta_a}$ consisting of fewer than

i intervals, impossible by the induction hypothesis. Thus i intervals are needed.

If $\bar{l}_a < \beta_a$, then in $U_{\bar{l}_a}$ the feasible point \bar{l} differs from the point $(\beta_1, \dots, \beta_{a-1}, \bar{l}_a)$ in $i - 1$ positions. Now suppose that there is a partition of $U \setminus F$ consisting of fewer than i intervals. The number of intervals in this partition must be $i - 1$ since a smaller number would imply an impossible partition of $U_{\bar{l}_a} \setminus F_{\bar{l}_a}$, as before. In fact, each set in the partition must intersect $U_{\bar{l}_a}$ yielding a non-empty interval that does not overlap with $F_{\bar{l}_a}$. Since all of $U \setminus F$ must be covered by this partition, some interval must contain a point in $U_{\bar{l}_a} \setminus F_{\bar{l}_a}$ and the point $(\alpha_1, \dots, \alpha_{a-1}, \beta_a)$, which lies “above” F , impossible by the definition of an interval. So i intervals are needed and the result is proved. \square

An important special case of Proposition 2.2 occurs when \bar{l} differs from the point β in only one position, in which case $U \setminus F$ is an interval. Since the symmetric result for infeasible sets holds, we have the following immediate consequence.

Corollary 2.1 *Let $\{I_j\}_{j \in \mathcal{A}}$ be a collection of infeasible sets, each of which is defined by an infeasible point that differs from the point α in only one position. Then $U \setminus (\cup_{j \in \mathcal{A}} I_j)$ is an interval.*

That is, as long as we restrict ourselves to infeasible sets whose defining points differ from α in only one position, we may remove arbitrary collections of infeasible sets in addition to the feasible set F without impacting the number of undetermined intervals produced. Of course, a symmetric result is implied for removing feasible sets along with an infeasible set. Thus Corollary 2.1 clearly applies to GSD as stated in Section 2.4. In fact, in view of (2.10) and (2.14), removing F from the reduced interval $U \setminus (\cup_{j \in \mathcal{A}} I_j)$ will likely result in fewer intervals. Thus we have proved the following key result.

Theorem 2.1 *In GSD, removing infeasible (feasible) sets from an undetermined set U in addition to the primary feasible (infeasible) set cannot increase the number of intervals needed to partition the remainder of U .*

Thus GSD is able to remove both types of sets simultaneously without splintering the remaining undetermined points into many sets. One might wonder whether this fact distinguishes GSD in any way, or if indeed *any* routine designed to remove feasible and infeasible sets would perform as well in this regard. The key to the answer lies in Theorem 2.2 below. (The cases in which $i^F \leq 1$ or $i^I \leq 1$ are already dealt with in Proposition 2.2.)

Theorem 2.2 *Let \bar{l}^F be a feasible point which differs from the point β in $i^F > 1$ positions, and let F be the corresponding feasible set. Let \bar{l}^I be an infeasible point which differs from the point α in $i^I > 1$ positions, and let I be the corresponding infeasible set. Then the number of sets needed to partition $U \setminus (F \cup I)$ is greater than or equal to the number of sets needed to partition $U \setminus F$.*

Proof: (By induction) If $a = 2$, then we clearly have $i^F = i^I = 2$. Already Proposition 2.2 provides that $U \setminus F$ needs 2 intervals for its partition (i.e., that $U \setminus F$ is an "L-shaped" lattice). Since $i^I = 2$ (i.e., \bar{l}^I lies strictly in the interior of $U \setminus F$), it is clear that $U \setminus (F \cup I)$ is not a single interval. Thus at least 2 intervals are needed.

Now suppose that the result holds for $a-1$ variables (X_1, \dots, X_{a-1}) , and consider the case of a variables (X_1, \dots, X_a) . As before, it is useful to think of U as being composed of ordered, indexed $(a-1)$ -dimensional undetermined sets $U_{\alpha_a}, \dots, U_{\beta_a}$. We consider two separate cases.

Case 1 ($\bar{l}_{j^*}^F \geq \bar{l}_{j^*}^I$ for some $j^* \in A$): Without loss of generality, assume that $j^* = a$. If $\bar{l}_a^F = \beta_a$, then also $\bar{l}_a^I = \beta_a$. In the $(a-1)$ -dimensional interval U_{β_a} , then, we have a feasible set $F_{\beta_a} = F \cap U_{\beta_a}$ whose defining point \bar{l}^F differs from β in i^F positions, and an infeasible set $I_{\beta_a} = I \cap U_{\beta_a}$. By the induction hypothesis, the partition of $U_{\beta_a} \setminus (F_{\beta_a} \cup I_{\beta_a})$ requires at least i^F intervals. Consequently, at least i^F are needed to partition the set $U \setminus (F \cup I)$.

Now, suppose that $\bar{l}_a^F < \beta_a$. $U_{\bar{l}_a^F}$ contains a feasible set $F_{\bar{l}_a^F} = F \cap U_{\bar{l}_a^F}$ whose defining point differs from $(\beta_1, \dots, \beta_{a-1}, \bar{l}_a^F)$ in $i^F - 1$ positions and an infeasible set $I_{\bar{l}_a^F}$. By the induction hypothesis, the partition of $U_{\bar{l}_a^F} \setminus (F_{\bar{l}_a^F} \cup I_{\bar{l}_a^F})$ requires at least $i^F -$

1 intervals. Note that, by the definition of an interval, none of these $i^F - 1$ intervals, even when extended to all levels of U , can contain the point $(\alpha_1, \dots, \alpha_{a-1}, \bar{l}_a^F)$, which lies "above" the set F . So more than these $i^F - 1$ intervals are needed to partition $U \setminus (F \cup I)$. This concludes this case.

Case 2 ($\bar{l}_j^F < \bar{l}_j^I$ for all $j \in \mathcal{A}$): In this case, necessarily $\bar{l}_j^F < \beta_j$ for all $j \in \mathcal{A}$, so that $i^F = a$. Similarly, $i^I = a$. Suppose that there exists a partition of $U \setminus (F \cup I)$ consisting of fewer than a intervals. This partition intersects the $(a-1)$ -dimensional interval U_{α_a} to form a partition of $U_{\alpha_a} \setminus F_{\alpha_a}$ (where $F_{\alpha_a} = F \cap U_{\alpha_a}$) consisting of fewer than a intervals. Now in U_{α_a} , the defining point of F_{α_a} differs from the point $(\beta_1, \dots, \beta_{a-1}, \alpha_a)$ in $a-1$ positions, and hence by Proposition 2.2 $a-1$ intervals are required to partition $U_{\alpha_a} \setminus F_{\alpha_a}$. Thus the partition of $U \setminus (F \cup I)$ requires $a-1$ intervals, each of which intersects U_{α_a} in a nonempty interval outside of the set F . This means that one of these intervals must contain points in $U_{\alpha_a} \setminus F_{\alpha_a}$ and the point $(\alpha_1, \dots, \alpha_{a-1}, \bar{l}_a^F + 1)$ (a point "above" F). This is not possible given the definition of an interval. Thus at least i^F intervals are needed to partition $U \setminus (F \cup I)$. This concludes this case and the result is now proved. \square

Thus we see that removing even a *single* infeasible set not of the form used in GSD can result in an increase in the number of sets needed to partition the remainder. This situation would only be compounded by the removal of multiple arbitrary infeasible sets. The following result gives an upper bound on the number of intervals needed to partition $U \setminus (F \cup I)$ in Theorem 2.2.

Proposition 2.3 *Let F and I be as in Theorem 2.2. Then $U \setminus (F \cup I)$ can be partitioned into $i^F + i^I - 1$ intervals.*

Proof: Let $j^* \in \mathcal{A}$ be such that $\bar{l}_{j^*}^F < \bar{l}_{j^*}^I$. (We must be able to find j^* since F and I cannot overlap.) Define

$$U_{j^*} = \{I \in U : \bar{l}_{j^*}^F + 1 \leq l_{j^*} \leq \bar{l}_{j^*}^I - 1 \\ \alpha_i \leq l_i \leq \beta_i \quad \text{for } i \neq j^* \}$$

(Note that $U_{j^*} = \emptyset$ if $\bar{l}_{j^*}^F = \bar{l}_{j^*}^I - 1$). What remains, then, is an interval

$$U_F = [\alpha, (\beta_1, \dots, \beta_{j^*-1}, \bar{l}_{j^*}^F, \beta_{j^*+1}, \dots, \beta_a)]$$

containing the feasible set F (whose defining point differs from U^F 's upper endpoint in $i^F - 1$ positions), and an interval

$$U_I = [(\alpha_1, \dots, \alpha_{j^*-1}, \bar{l}_{j^*}^I, \alpha_{j^*+1}, \dots, \alpha_a), \beta]$$

containing the infeasible set I (whose defining point differs from U^I 's lower endpoint in $i^I - 1$ positions). By Proposition 2.2, we partition U_F using $i^F - 1$ sets and U_I using $i^I - 1$ sets. Thus, counting U_{j^*} , we have used $(i^F - 1) + (i^I - 1) + 1 = i^F + i^I - 1$ intervals in the decomposition. \square

To summarize, our goal in this section was to investigate the consequences of removing infeasible sets in addition to a primary feasible set (or feasible sets in addition to a primary infeasible set). In GSD, such an action causes the number of intervals needed for the partition either to stay the same or to decrease. If the secondary sets are not of the form used in GSD (i.e., if they are infeasible sets whose defining points differ from α in more than one position or if they are feasible sets whose defining points differ from β in more than one position), the number needed either stays the same or increases. This brings us to the following result.

Corollary 2.2 *Let \mathcal{D} be the class of all decomposition schemes that consist of removing a feasible (infeasible) interval and one or more infeasible (feasible) intervals from an undetermined interval U using Proposition 2.1. Then GSD is minimal in \mathcal{D} with respect to the number of new undetermined sets produced per iteration.*

One final point must be made. Each decomposition scheme in \mathcal{D} involves a variety of complex interactions that determine its ultimate effectiveness on a given problem. Experience indicates that no single objective (e.g., keeping the list of sets short, removing maximum probability, expending little computational effort) can guarantee across-the-board success. Trying to file few sets *does* seem to play a

big role, and intuitively contributes in avoiding the splintering of a set into many small subsets. Nevertheless, because of the complex interplay mentioned above, we shall attempt in Chapters 3 and 4 to remove a single arbitrary infeasible set and an arbitrary feasible set in each iteration, ignoring the warning of Corollary 2.2. This “hybrid” approach will be compared with pure and mixed GSD applications for the stochastic minimum spanning tree and minimum cost flow problems.

In the final sections of this chapter, we conclude our description of GSD by detailing its use for producing bounds and providing input for variance-reducing Monte Carlo estimation schemes.

2.6 Using GSD to Produce Bounds

In Section 2.4 we presented an algorithm for computing P^F exactly in a number of iterations that is typically small (relative to $|\Omega|$) for moderate size problems. For large problems, this may still represent an unacceptable computational workload. In such cases the real strength of a routine like GSD lies in its ability to generate information after each iteration which can be used for computing bounds, designing efficient Monte Carlo sampling plans, and conducting sensitivity analysis with respect to changing input distributions [2],[4]. We have pointed out that GSD has available at all times a lower bound for P^F . In this section, we briefly discuss how we can stop short of complete decomposition of Ω and retrieve both lower and upper bounds for P^F .

Suppose we carry out a number of iterations of GSD, resulting in accumulated feasible probability P_l^F and a collection of m remaining disjoint undetermined intervals which we may renumber as $\{U_i\}_{i=1}^m$. Given the way it was accumulated, it is clear that P_l^F is a lower bound for P^F . Since all feasible points not yet accounted for are contained in $\cup_{i=1}^m U_i$, we have

$$P_l^F \leq P^F \leq P_l^F + \sum_{i=1}^m P\{U_i\} \equiv P_u^F. \quad (2.15)$$

That is, stopping the GSD procedure at any point yields lower and upper bounds on the probability of interest. Furthermore, since nearly every iteration removes some undetermined probability (by identifying feasible and infeasible sets) and adds to the accumulated feasible probability, the sequence of lower bounds is nondecreasing and the sequence of upper bounds is nonincreasing. The decomposition may then be terminated based on one of the following stopping conditions:

- Stop when a desired number of sets has been decomposed.
- Stop when bounds are sufficiently close.

We may alter the procedure GSD above by substituting step $\tilde{4}$ for step 4 and adding steps 5 and 6:

Step $\tilde{4}$ If $\mathcal{U} = \emptyset$, stop. If a desired stopping condition is met, go to step 5. Otherwise, remove a set U from \mathcal{U} and go to step 2.

Step 5 Let $P_u^F = P_l^F$.

Step 6 For $i = 1, \dots, m$: Evaluate extremes of U_i . If U_i is infeasible, discard U_i . If U_i is feasible, set $P_l^F = P_l^F + P\{U_i\}$ and $P_u^F = P_u^F + P\{U_i\}$. Otherwise, since U_i contains some feasible points but is not completely feasible, set $P_u^F = P_u^F + P\{U_i\}$.

(In step 6, we may skip the evaluation of the extremes and simply set $P_u^F = P_u^F + P\{U_i\}$ if evaluating the extremes is deemed computationally expensive.) After the revised GSD has been performed, we are left with lower bound P_l^F and upper bound P_u^F , and with a set of $\tilde{m} \leq m$ undetermined intervals. These bounds and remaining undetermined sets may be used as input to Monte Carlo sampling plans for estimating P^F , as described in [1], [2] and presented in the following section.

The revised GSD is important since in many cases, deriving a set of tight bounds is just as useful as evaluating P^F exactly (especially for large problems, in which it is the only practical recourse). In fact, it is the desire for quick, tight bounds that

motivated some of the features of GSD. For example, filing and retrieving sets by probability is an attempt to ensure that no high-probability sets remain undetermined (an important goal since bounds can differ by as much as $\sum_{i=1}^m P\{U_i\}$). Some decomposition approaches do not identify limiting indices \hat{l}_j so that only a feasible set and undetermined sets are produced in each iteration. Thus the upper bound is reduced only by intervals which are found to be completely infeasible. When limiting indices are used, both bounds may improve in nearly every iteration. Finally, no existing decomposition approach decomposes both *feasibly and infeasibly*. We anticipate that for some problems this added flexibility will allow as much tightening of bounds as possible from both directions, with benefits exceeding the cost of the additional computational effort.

2.7 GSD Bounds as Input to Monte Carlo Estimation Routines

We may take the *partial decomposition* results of Section 2.6 and use them as input to a highly efficient Monte Carlo sampling routine. The sampling plan described in this section combines importance and stratified sampling, requires no more work per iteration than a crude Monte Carlo sampling of Ω , and has much smaller variance than such a plan. In fact, the ratio of crude Monte Carlo variance to the variance of this sampling plan based on GSD output is at least

$$1 / \left[\sqrt{P_u^F(1 - P_l^F)} - \sqrt{P_l^F(1 - P_u^F)} \right]^2, \quad (2.16)$$

which can be computed before sampling begins.

We begin with bounds P_l^F and P_u^F , and \tilde{m} remaining undetermined intervals $\{U_i\}_{i=1}^{\tilde{m}}$. Let $\pi_i = P\{U_i\}$ for all i and let $\pi = \sum_{i=1}^{\tilde{m}} \pi_i$. For $l \in \Omega$, define $\phi(l) = 1$ if l is a feasible point, $\phi(l) = 0$ if l is infeasible. Suppose we wish to draw N independent samples from the undetermined sets. For $i = 1, \dots, \tilde{m}$ we shall draw $N_i = N\pi_i/\pi$ samples from set U_i as follows.

For $n = 1, \dots, N_i$:

- For each $j = 1, \dots, a$: sample a level l_j according to the distribution

$$\left\{ \frac{p_j(l)}{\sum_{i=\alpha_j}^{\beta_j} p_j(i)}, \quad l \in \{\alpha_j, \dots, \beta_j\} \right\}. \quad (2.17)$$

- Form the resulting point $\mathbf{X}^{(n)} = (l_1, \dots, l_a)$ and evaluate $\phi(\mathbf{X}^{(n)})$.

Then

$$\hat{P}^F = P_i^F + \sum_{i=1}^{\hat{m}} \pi_i \left[\sum_{n=1}^{N_i} \phi(\mathbf{X}^{(n)}) / N_i \right] \quad (2.18)$$

is an unbiased estimator of P^F with variance as given in [1].

This efficient sampling routine is easy to implement, as the U_i 's may be taken one at a time, used for sampling and then discarded. It adds even greater flexibility to the GSD process, especially for large problems.

2.8 Conclusions

In this chapter, we have presented the General State Space Decomposition algorithm, a powerful and flexible tool for evaluating measures related to the operation of stochastic systems. Starting with the significant groundwork laid by previous studies, GSD has strengthened the original decomposition of Doulliez and Jamouille in the following ways:

- 1) The general framework and vocabulary developed in Section 2.2 give a common point of departure for discussing and comparing the application of these types of decompositions. The GSD routine itself represents an advancement as an effective application of factoring that is general enough to be easily tailored to address a variety of stochastic problems.

- 2) The content of Section 2.5 is important in establishing GSD as best (in minimizing the number of undetermined sets produced in each iteration) among similar alternative approaches.
- 3) Previous decomposition approaches employed either Pure Strategy 1 or Pure Strategy 2 (see Section 2.4). Such strategies will likely be good for some choices of constraints, bad for others (see Section 3.4). GSD as applied in Chapter 3 represents the first routine of this type which allows mixed strategies, thus attempting to gain the benefits of both feasible and infeasible decomposition when possible.
- 4) The use of GSD both in Chapter 3 and in Chapter 4 continues to demonstrate the importance of efficient list processing for \mathcal{U} , being one of the first such applications to keep the undetermined sets in a sorted heap structure (see also [2]).

CHAPTER 3

The Stochastic Minimum Spanning Tree Problem

3.1 Introduction

Consider an undirected, connected graph $G = (\mathcal{N}, \mathcal{A})$ with node set $\mathcal{N} = \{1, \dots, n\}$ and arc set $\mathcal{A} = \{1, \dots, a\}$. A *spanning tree* is a maximally acyclic connected subgraph of G , so named because it reaches every node in \mathcal{N} . Suppose that each arc in \mathcal{A} has an associated nonnegative weight. An important problem in network optimization is the identification of a spanning tree with the smallest possible total weight. This so-called *minimum spanning tree* (MST) problem is important in its own right as it attempts to connect all nodes in, e.g., a communications network as economically as possible. In addition, there are other unrelated network optimization problems that can be cast as MST problems. A number of efficient algorithms exist for solving the minimum spanning tree problem (see [24],[28],[29]).

In this chapter, we are concerned with describing the probabilistic behavior of MSTs in networks whose arcs have random finite weights. Specifically, the weight of arc $j \in \mathcal{A}$ is a discrete random variable X_j that takes on weight values $w_j(1) < \dots < w_j(n_j)$ with respective probabilities $p_j(1), \dots, p_j(n_j)$. We assume that the components of $\mathbf{X} = (X_1, \dots, X_a)$ are mutually independent. The state space Ω of \mathbf{X} contains $|\Omega| = \prod_{j=1}^a n_j$ elements of the form $x = (w_1(l_1), \dots, w_a(l_a))$, where $l_j \in \{1, \dots, n_j\}$ for each j . As in Section 2.2, we employ the convention of writing this state point as $l = (l_1, \dots, l_a)$.

We shall concern ourselves in this chapter with the following problems related to MSTs in stochastic networks.

Problem ST1 Define the random variable $W(\mathbf{X})$ as the weight of a MST and let $d \geq 0$. We want to compute the probability $P\{W(\mathbf{X}) \leq d\}$ that we can construct a spanning tree whose weight does not exceed d .

For this problem a realization l of \mathbf{X} is considered feasible if $W(l) \leq d$, and clearly X_1, \dots, X_a are all lower feasible. This problem will be discussed in Section 3.2, including a modification to GSD that will allow the simultaneous determination of the entire distribution of $W(\mathbf{X})$.

Problem ST2 Let $e \in \mathcal{A}$. We want to compute the probability that e belongs to a minimum spanning tree.

Associated with every realization l of \mathbf{X} is a collection $\mathcal{T}(l) = \{T_1, \dots, T_m\}$ of minimum spanning trees. The state point l is feasible if $e \in T_i$ for some $i \in \{1, \dots, m\}$. For this problem, the random variable X_e is lower feasible while the weight of all other arcs is upper feasible. This problem, along with two modifications, will be discussed in Section 3.3.

Before solving these problems by using GSD, it will be instructive to make some observations about the dynamics of MSTs when an arc changes weight. These observations will be useful since each GSD procedure will consist of a sequence of steps in which the level of a *single* arc is changed. By knowing in advance the consequences of these steps, we can maintain or achieve properties such as tree membership or tree weight.

We begin at an arbitrary point $l = (l_1, \dots, l_a)$. Let T be a MST at this level. We shall independently decrease and increase the weight of arcs both on and off the tree T addressing the following questions for each action:

- 1) Does this action cause T to cease being a MST, thus forcing a change in tree membership?

2) Does this action change the weight of a MST?

We summarize these observations below as Properties 1-4. For convenience, we shall use a shorthand notation for some set operations. For example, the operation $T \cup \{i\} \setminus \{j\}$ will be written as $T + i - j$.

Property 1 (Decreasing the weight of an arc $j \in T$) Since T is a MST with the current arc weights, it will surely remain so if the weight of one of its arcs decreases. Of course, the weight of T will decrease by the amount of the reduction in the weight of arc j .

Property 2 (Decreasing the weight of an arc $j \notin T$) If $j \notin T$, then decreasing its weight sufficiently can cause it to displace from T an arc with greater weight. Thus we must first investigate $C(j)$, the unique cycle that arc j forms with T . Only arc i , the most heavily weighted arc in $C(j) - j$, can be so displaced in order to obtain a minimum spanning tree. As long as the weight of arc j remains at least as large as that of arc i , the reduction of j impacts neither T nor its weight. Otherwise, $T - i + j$ is the new MST, with weight less than that of T .

Property 3 (Increasing the weight of an arc $j \notin T$) Since $j \notin T$ with current weights, the weight of arc j can be increased arbitrarily without impacting T or its weight.

Property 4 (Increasing the weight of an arc $j \in T$) Increasing the weight of j can cause T to cease being a MST. Among those arcs not on T whose cycles with T contain j , let i be the arc with the smallest weight. As long as the weight of arc j does not exceed that of arc i , T is still a MST (though with increased weight). Otherwise, $T - j + i$ is a MST and further increases in the weight of j are governed by Property 3.

We shall often refer to the process of exchanging arcs between a MST and the remainder of \mathcal{A} as *pivoting*.

We now proceed to the solution of Problems ST1 and ST2 in Sections 3.2 and 3.3, respectively. Section 3.4 contains numerical examples. In Section 3.5 we suggest extensions of the MST results to other matroid settings. Concluding remarks are made in Section 3.6.

3.2 Computing the Probability Distribution of the Weight of a MST

We consider here the stochastic network system that operates feasibly if the constraint $W(\mathbf{X}) \leq d$ is satisfied, and seek to compute the probability $P\{W(\mathbf{X}) \leq d\}$. Unfortunately this problem, like many that fit the general problem description in Section 2.3, is provably hard. We show this by reduction from the following #P-complete problem (see [6]).

All-terminal Undirected Rational Network Reliability Problem (ATRP)

Inputs: Undirected network with arcs that function or fail independently of each other. For arc i , the probability of failure is b_i/c_i , where b_i is a non-negative integer and c_i is a positive integer such that $b_i \leq c_i$.

Outputs: b and c , where c is a positive integer, b is a non-negative integer such that $b \leq c$, and $P\{\text{the system functions}\} = P\{\text{the network is connected}\} = P\{\text{there is a spanning tree consisting of functioning arcs}\} = b/c$.

Proposition 3.1 *The evaluation of $P\{W(\mathbf{X}) \leq d\}$ is #P-hard.*

Proof: Consider an instance of ATRP. To each arc i , assign the random weight

$$X_i = \begin{cases} 0 & \text{with probability } 1 - \frac{b_i}{c_i} \\ 1 & \text{with probability } \frac{b_i}{c_i} \end{cases}$$

where nonzero weight corresponds to failure. Then evaluating $P\{W(\mathbf{X}) \leq 0\}$ is equivalent to finding the probability that there is a spanning tree made up of functioning arcs. That is, $P\{W(\mathbf{X}) \leq 0\}$ is equal to the all-terminal network reliability. Thus, the ability to evaluate $P\{W(\mathbf{X}) \leq d\}$ implies the ability to solve ATRP. Since ATRP is #P-complete, the evaluation of $P\{W(\mathbf{X}) \leq d\}$ is #P-hard. \square

As discussed earlier, all arc weight random variables are lower feasible. The extreme feasible candidate of an undetermined set $U = [\alpha, \beta]$ is simply the point α and the extreme infeasible candidate is β .

We are now in a position to discuss the application of GSD to the present problem. Given the groundwork laid in Chapter 2, this will amount to little more than specifying the mechanics of finding \bar{l} and the \hat{l}_j 's for feasible and infeasible decomposition. As remarked in Chapter 2, there are several ways to describe feasible and infeasible decomposition for this problem. In Section 3.2.1 we present the end-product of a long evolution of decomposition schemes. It is fairly involved, but has shown itself to be a flexible routine. Section 3.2.2 contains an alternative algorithm which is conceptually much simpler, and which out-performs the more sophisticated algorithm of Section 3.2.1 in some instances. Section 3.2.3 contains a "hybrid" routine based on the decomposition discussed in the proof of Proposition 2.3. Finally, Section 3.2.4 presents a modification to the work in Section 3.2.1 that allows us to compute the entire distribution of $W(\mathbf{X})$ simultaneously.

3.2.1 Spanning Tree Decomposition Algorithm I

The algorithm presented in this section is designed to carry out both feasible and infeasible decomposition, as outlined in Chapter 2. It can be executed in several ways, namely using Pure Strategies 1 and 2 and Mixed Strategies 1 through 3, as described in Section 2.4. All five options will be used to solve the sample problems of Section 3.4.

We now give the mechanics of Spanning Tree Decomposition Algorithm I (STDA-I) feasible decomposition and of STDA-I infeasible decomposition. In both cases we

describe a single iteration, in which we partition an undetermined interval $U = [\alpha, \beta]$, which contains feasible and infeasible states.

STDA-I Feasible Decomposition

To determine the feasible point \bar{l} , we begin at the extreme feasible point α and increase arc weights while remaining feasible. For all $j \in \mathcal{A}$, we set the arc weight at $w_j(\alpha_j)$ and find a MST, which we denote $T(\alpha)$. It is clear by Property 3 that any j not in $T(\alpha)$ can be raised to weight $w_j(\beta_j)$ without impacting $T(\alpha)$. Thus with no additional work we may set

$$\bar{l}_j = \begin{cases} \beta_j & \text{if } j \notin T(\alpha) \\ \alpha_j & \text{if } j \in T(\alpha) \end{cases}$$

yielding a feasible point for which $a - n + 1$ arcs are cut as deeply as possible. At this point we could try to increase the weight of some of the arcs on the tree $T(\alpha)$ using Property 4. Experimentation has indicated that the considerable extra work that would be involved does not pay off in this case. Therefore, we shall use \bar{l} as given above.

All that remains, then, is to determine the limiting feasible indices \hat{l}_j . We already have determined $a - n + 1$ of them: since $\hat{l}_j \geq \bar{l}_j$ for all j , our assignment of \bar{l} above indicates that $\hat{l}_j = \beta_j$ for all $j \notin T(\alpha)$. Increasing the weight of the tree arcs requires, according to Property 4, some pre-processing to determine what restrictions there may be on such an action. Fortunately, though, since each \hat{l}_j is determined independently and starting at the same point α , all the tree arcs can be preprocessed in one pass through the arcs off $T(\alpha)$ by the following procedure.

Procedure SHORT

Step 1 For $j \in T(\alpha)$: $SHORT(j) = \infty$.

Step 2 For $j \notin T(\alpha)$: Find $C(j)$, the unique cycle j forms with $T(\alpha)$. For all $i \in C(j) - j$, put $SHORT(i) = \min\{SHORT(i), w_j(\alpha_j)\}$.

Step 3 End.

For all $j \in T(\alpha)$ this procedure produces $SHORT(j)$, the weight of the off-tree arc that would displace arc j if the weight of this arc were to increase beyond $SHORT(j)$. For $j \notin T(\alpha)$ we set $\hat{l}_j = \beta_j$. We then proceed to find \hat{l}_j independently for each $j \in T(\alpha)$ as follows. We begin with the MST $T(\alpha)$ of weight $W(\alpha)$ and all arcs at levels α (this information having been saved from the determination of \bar{l}). If $\bar{l}_j = \beta_j$, we set $\hat{l}_j = \beta_j$ and go to the next arc. Otherwise, we set $\hat{l}_j = \max\{l : \alpha_j \leq l \leq \beta_j, w_j(l) \leq SHORT(j)\}$.

The determination of \hat{l} involves relatively little work. Procedure SHORT takes time $O(an)$, and the subsequent determination of the \hat{l}_j 's (which is done for the tree arcs only) takes time $O(n * \max_{j \in A} n_j)$.

It might seem inconsistent to the reader that in finding \bar{l} we declined to raise any tree arcs so as to avoid costly preprocessing, yet we did the preprocessing for the \hat{l}_j 's. The reason for this apparent inconsistency is that these situations differ in two important ways. First, with the \hat{l}_j 's one preprocessing pass through the off-tree arcs was sufficient to set up all \hat{l}_j determinations because the weight increases used to find the \hat{l}_j 's were independent and started at the same point α . With \bar{l} , since the point is determined considering all arcs jointly, none of the weight increases are independent. And since in general each arc is raised from a different state point (wherever the previous arc left off), we would have had to execute SHORT as many as $n - 2$ times (the preprocessing for the first tree arc raised was done in finding \hat{l}). Secondly, as mentioned in Section 2.4, \hat{l}_j must be pushed as far as possible, so that there was no choice but to preprocess.

After limiting feasible indices have been determined for all arcs, we are ready to decompose as in Section 2.4 and proceed to the next undetermined interval.

STDA-I Infeasible Decomposition

Infeasible decomposition begins at the extreme point β (which by pre-screening we know to be an infeasible point) and pushes in, identifying an infeasible point \bar{l}

and limiting infeasible indices \hat{l}_j . As before, the determination of \hat{l}_j for each j and the determination of \bar{l} , each beginning at β , can be designed to be carried out in any order. In feasible decomposition, we found \bar{l} first. Here, the limiting infeasible indices \hat{l}_j will be found first. Afterward, we use the fact that $\hat{l}_j \leq \bar{l}_j$ for $j \in \mathcal{A}$ during the derivation of \bar{l} .

We begin by finding $T(\beta)$, a MST at the extreme infeasible point, with weight $W(\beta)$. For $j \in T(\beta)$ we can use Property 1 to find \hat{l}_j with relative ease, since we can look at arc j in isolation, ignoring all other arcs. We initially set $\hat{l}_j = \beta_j$ and lower one level at a time until either the MST weight would drop below d with further reduction or \hat{l}_j hits α_j while the tree weight is still infeasible.

On the other hand, finding \hat{l}_j for off-tree arcs j is complicated by the fact that reducing the weight of arc j could force it into the MST, as discussed above in Property 2. So we identify $C(j)$, the unique cycle that arc j forms with $T(\beta)$, and identify i , the most heavily weighted arc (at level β) in $C(j) - j$. We then decrease the weight of arc j one level at a time until either we hit α_j and still have $w_j(\alpha_j) \geq w_i(\beta_i)$, in which case $\hat{l}_j = \alpha_j$ (here we also preset $\bar{l}_j = \alpha_j$), or at some point the weight of arc j becomes strictly less than that of arc i and $T(\beta) - i + j$ is the new MST (here we preset \bar{l}_j at the index prior to that which forced j to enter the tree). If this tree weight is still feasible, we continue reducing the weight of arc j as for tree arcs. If not, then \hat{l}_j is set at the level just prior to the level at which arc j entered the tree.

It remains to derive the infeasible point \bar{l} by reducing weights from the level β . Rather than starting from $\bar{l} = \beta$, we begin with the point given by

$$\bar{l}_j = \begin{cases} \beta_j & \text{if } j \in T(\beta) \\ \text{as set above} & \text{if } j \notin T(\beta) \end{cases} \quad (3.1)$$

It is clear, by Properties 1 through 4, that $T(\beta)$ is a MST for \bar{l} defined in this way, and that $W(\beta)$ is its weight. We may proceed from this point to derive a deeper infeasible point. As mentioned earlier, $\bar{l}_j \geq \hat{l}_j$ for all $j \in \mathcal{A}$. Thus the state \bar{l} serves

as a floor for the further reduction of \bar{l} . In fact, it is likely that for some arcs j we already have $\bar{l}_j = \hat{l}_j$ with \bar{l} as defined by (3.1). Those arcs will not be considered for further reduction. The remaining arcs will be reduced one at a time as much as possible, governed by the applicable Property, as long as the tree weight does not reach d .

This further reduction could be accomplished in many different ways. The idea is to reduce as many arcs as possible, and to reduce them as much as possible. We want to avoid using arcs $j \in T(\beta)$ for which $w_j(\bar{l}_j) - w_j(\bar{l}_j - 1)$ is large since such arcs might bring the MST weight down to d all at once, preventing the reduction of other arcs. In addition, we would like to give preference to arcs with a lot of room $\bar{l}_j - \hat{l}_j$ for reduction. In order to try to accomodate both goals, we use an ordering of arcs that is inspired by a knapsack heuristic. (If $n_j = 2$ for all j , we solve a 0-1 knapsack problem. The space taken by each arc in that case is $w_j(\beta_j) - w_j(\alpha_j)$, all arcs have equal value, and the available space in the knapsack is $W(\beta) - d$.) Specifically, after eliminating arcs with $\bar{l}_j = \hat{l}_j$ from consideration, we first consider tree arcs, followed by arcs not on $T(\beta)$, in decreasing order of the ratio $(\bar{l}_j - \hat{l}_j) / [w_j(\bar{l}_j) - w_j(\bar{l}_j - 1)]$. The weight of each arc selected for reduction according to this rule is decreased one level at a time until further reduction either is not possible (i.e., $\bar{l}_j = \hat{l}_j$) or would result in an infeasible point.

Once \bar{l} has been determined, we are ready to decompose as in Section 2.4 and proceed to the next undetermined interval.

3.2.2 Spanning Tree Decomposition Algorithm II

Section 3.2.1 gave one possible application of GSD to the determination of an ordinate of the probability distribution of minimum spanning tree weights. As mentioned before, there is nearly limitless flexibility in applying GSD to a given problem instance. STDA-I, given in Section 3.2.1, contains what seem to be the most effective features of a number of earlier schemes. This section describes Spanning Tree Decomposition Algorithm II (STDA-II), an algorithm that is conceptually much

simpler. We include it because it is fast and because it demonstrates the flexibility of GSD by its departure from the routine described in Chapter 2. Differences include a modified prescreening phase and a simplified decomposition which results from not finding limiting indices \hat{l}_j . Also, except for undetermined intervals ruled infeasible in prescreening, no sets will be decomposed infeasibly.

Before describing STDA-II we can simplify the descriptions given earlier that define the partitioning of an undetermined interval. Since no variables are upper feasible and no limiting indices are found, we have simply

$$F = \{l \in U : \alpha_j \leq l_j \leq \bar{l}_j \text{ for } j \in \mathcal{A}\} \quad (3.2)$$

and

$$\begin{aligned} U_j = \{l \in U : & \alpha_i \leq l_i \leq \bar{l}_i \quad \text{for } i < j \\ & \bar{l}_j + 1 \leq l_j \leq \beta_j \\ & \alpha_i \leq l_i \leq \beta_i \quad \text{for } i > j\}. \end{aligned} \quad (3.3)$$

STDA-II can be stated as follows, with explanation following.

Procedure STDA-II

Step 1 Start with $U = \Omega$. Set $\mathcal{U} = \emptyset$, $P_l^F = 0$.

Step 2 Let α and β denote the lower and upper endpoints of U , respectively. Find $T(\alpha)$, a MST at α , with weight $W(\alpha)$. If $W(\alpha) > d$, then U is infeasible; go to step 7.

Step 3 Let $W_\beta(\alpha) = \sum_{j \in T(\alpha)} w_j(\beta_j)$. If $W_\beta(\alpha) \leq d$, then $W(\beta) \leq d$ and U is feasible. Set $P_l^F = P_l^F + P\{U\}$ and go to step 7.

Step 4 Set $\bar{l}_j = \beta_j$ for $j \notin T(\alpha)$, $\bar{l}_j = \alpha_j$ for $j \in T(\alpha)$, $STWT = W(\alpha)$. Sort the arcs $j \in T(\alpha)$ for which $\alpha_j < \beta_j$ in increasing order of $w_j(\beta_j) - w_j(\alpha_j)$.

Step 5 Taking the tree arcs in this order, do for j chosen: Set $\bar{l}_j = \beta_j$, $STWT = STWT + w_j(\beta_j) - w_j(\alpha_j)$. As long as $STWT \leq d$, continue choosing arcs. When arc j causes $STWT > d$, put

$$\bar{l}_j = \max\{\gamma : \alpha_j \leq \gamma \leq \beta_j, STWT - w_j(\beta_j) + w_j(\gamma) \leq d\} \quad (3.4)$$

and go to step 6.

Step 6 Form F as in (3.2) and set $P_l^F = P_l^F + P\{F\}$. For $j \in A$: If $\bar{l}_j \neq \beta_j$, file U_j in \mathcal{U} , where U_j is given by (3.3).

Step 7 If $\mathcal{U} = \emptyset$, stop. Otherwise, remove a set U from \mathcal{U} and go to step 2.

This procedure can be altered in the usual way to produce bounds (See Section 2.6).

A few comments need to be made about STDA-II. We first look at Step 3. When the weights of the arcs are raised to the levels β , the tree $T(\alpha)$ will not in general be a minimum spanning tree. However, if its weight $W_\beta(\alpha)$ does not exceed d , then by definition neither will $W(\beta)$, the weight of a minimum spanning tree at β . Thus in such a situation we may declare U feasible without actually making a MST evaluation at β .

If $W_\beta(\alpha) > d$, we cannot conclude that β is an infeasible point. But we do know that there is a point with $\bar{l}_j = \beta_j$ for $j \notin T(\alpha)$, $\bar{l}_j < \beta_j$ for at least one $j \in T(\alpha)$, at which $T(\alpha)$ has feasible length. (The point with $\bar{l}_j = \alpha_j$ for all $j \in T(\alpha)$ is clearly such a point since necessarily $W(\alpha) \leq d$.) Consistent with our usual goal of keeping the list of undetermined sets small, we select from among the (potentially) many such points \bar{l} one which minimizes the number of new undetermined intervals produced. In view of (3.3), $U_j = \emptyset$ only if $\bar{l}_j = \beta_j$ and it is thus reasonable to choose a point \bar{l} for which $\bar{l}_j = \beta_j$ for as many $j \in T(\alpha)$ as possible. Thus we have a 0-1 knapsack problem to be solved. The available space in the knapsack is $d - W(\alpha)$. Each arc $j \in T(\alpha)$ takes up space equal to $w_j(\beta_j) - w_j(\alpha_j)$ (here putting arc j into the knapsack means setting $\bar{l}_j = \beta_j$) and all arcs have equal value, say 1. Steps 4 and 5 apply a standard knapsack heuristic, which is optimal in this case as shown

below. When the first arc is found that "does not fit," we increase its weight as much as possible from the level α_j (i.e., "put in" as much of it as possible) and leave the remaining tree arcs at the level α .

The following proposition shows that the steps 4 and 5 in procedure STDA-II result in a maximum number of arcs in $T(\alpha)$ with $\bar{l}_j = \beta_j$, thus producing a minimum number of new undetermined sets.

Proposition 3.2 *Steps 4 and 5 in procedure STDA-II yield a minimum number of new undetermined intervals in each iteration.*

Proof: We establish this result by showing that the heuristic knapsack solution produced by these steps is optimal. Note that for this problem all items considered for inclusion in the knapsack (the arcs in $T(\alpha)$) have equal value. Thus, the standard heuristic merely takes items in increasing order of $w_j(\beta_j) - w_j(\alpha_j)$.

Consider an optimal solution with value n° . If there is any arc $j \in T(\alpha)$ not included whose weight $w_j(\beta_j) - w_j(\alpha_j)$ is less than that of any arc in the knapsack, these arcs can clearly be exchanged to yield a solution that is still feasible and which also has value n° . That is, this produces an alternate optimal solution. We may repeat this procedure until all arcs in the knapsack have weights at least as small as those of arcs not included, again yielding an optimal solution. But this solution is precisely the solution gotten by applying the above heuristic. This proves the result. \square

Finally, since only those arcs $j \in T(\alpha)$ for which $\alpha_j < \beta_j$ are considered in Steps 4 and 5, we would like $T(\alpha)$ to contain as many arcs as possible for which $\alpha_j = \beta_j$ (or even $\alpha_j + 1 = \beta_j$). Thus we preprocess the arcs prior to finding $T(\alpha)$:

$$w(j) = \begin{cases} w_j(\alpha_j) - 2\epsilon & \text{if } \alpha_j = \beta_j \\ w_j(\alpha_j) - \epsilon & \text{if } \alpha_j + 1 = \beta_j \\ w_j(\alpha_j) & \text{otherwise} \end{cases} ,$$

where ϵ is suitably small. In this way as many of these preferred arcs as possible are

included. After finding $T(\alpha)$, the true arc weights are restored so that tree weights may be accurately computed.

STDA-II is an intuitively pleasing algorithm since it uses easily derived points and attempts to keep the list of undetermined sets small. It has shown promise for fast bound production. It will be tested on several examples in Section 3.4 along with the various forms of STDA-I and ST-HYBRID.

3.2.3 Hybrid Spanning Tree Decomposition Algorithm

Thus far we have addressed Problem ST1 using GSD procedures. We describe here ST-HYBRID, an algorithm that in each iteration decomposes an undetermined set $U = [\alpha, \beta]$ into one feasible set F , one infeasible set I and up to $2a - 1$ new undetermined sets. Its name stems from the fact that it is essentially a combination of STDA-I feasible decomposition and STDA-I infeasible decomposition. The following description summarizes this procedure.

Procedure ST-HYBRID

Step 1 Start with $U = \Omega$. Set $\mathcal{U} = \emptyset$, $P_l^F = 0$.

Step 2 Let α and β denote, respectively, the lower and upper endpoints of U . Find $T(\beta)$, a MST at β , with weight $W(\beta)$. If $W(\beta) \leq d$ then U is feasible. Set $P_l^F = P_l^F + P\{U\}$ and go to step 7.

Step 3 Find $T(\alpha)$, a MST at α , with weight $W(\alpha)$. If $W(\alpha) > d$ then U is infeasible. Go to step 7.

Step 4 Derive a feasible point \bar{l}^F based on the MST $T(\alpha)$ as in STDA-I feasible decomposition.

Step 5 Derive an infeasible point \bar{l}^I based on the MST $T(\beta)$ as in STDA-I infeasible decomposition.

Step 6 Set $P_I^F = P_I^F + P\{F\}$, where F is defined by (3.2). Decompose the remainder based on \bar{l}^F and \bar{l}^I as described in the proof of Proposition 2.3. File new undetermined intervals in \mathcal{U} .

Step 7 If $\mathcal{U} = \emptyset$, stop. Otherwise remove a set U from \mathcal{U} and go to step 2.

This procedure can be altered as in Section 2.6 to produce bounds.

Section 2.5 showed that routines like ST-HYBRID can file unnecessarily large numbers of new undetermined sets (compared with strict GSD routines). It was also pointed out, though, that there are many factors that jointly determine the ultimate performance characteristics of a decomposition routine. In Section 3.4 ST-HYBRID will be compared with the other routines in Section 3.2 on some sample problems.

We close this discussion of Problem ST1 with the following GSD modification that simultaneously determines the entire distribution of $W(\mathbf{X})$.

3.2.4 Computing the Entire Distribution of the Minimum Spanning Tree Weight

The previous sections have given decomposition algorithms for computing a single ordinate of the cumulative distribution function of the weight of a MST. Calculating $P\{W(\mathbf{X}) \leq d\}$ is sufficient for situations in which d represents an existing constraint on MST weight (e.g., a capital or material constraint). But when that is not the case, it is instructive to explore the full range of possible MST weights, calculating associated probabilities and computing the mean and higher moments of $W(\mathbf{X})$. In that case, we could execute STDA-I and/or STDA-II repeatedly for all possible MST weights between $W(1, \dots, 1)$ and $W(n_1, \dots, n_a)$. This would clearly involve a great deal of computational effort, particularly when there are a large number of possible realizations of $W(\mathbf{X})$. In this section we present a GSD modification that simultaneously determines the entire distribution for $W(\mathbf{X})$ in a single decomposition procedure. Procedure STDIST below computes the pmf $q(d) = P\{W(\mathbf{X}) = d\}$. Our objective requires a modification to the definition of

a feasible set. For each undetermined interval, the feasible states will have MST weights equal to that at the state α . (Clearly all variables are lower feasible.) We begin by finding $T(\alpha)$, a MST with arc weights corresponding to the point α . Properties 3 and 4 yield the feasible interval defined by the point

$$\bar{l}_j = \begin{cases} \alpha_j & \text{if } j \in T(\alpha) \\ \beta_j & \text{if } j \notin T(\alpha) \end{cases} \quad (3.5)$$

Up to a undetermined sets also result from this partition, using (3.3) in Section 3.2.2. This procedure is summarized below. The description assumes integer arc weights. Modifications would need to be made only in the data structures for the $q(d)$'s and the manner in which they are accumulated if arc weights are non-integral.

Procedure STDIST

Step 1 Find $W(1, \dots, 1)$, $W(n_1, \dots, n_a)$. For $d = W(1, \dots, 1), \dots, W(n_1, \dots, n_a)$, set $q_l(d) = 0$. Set $\mathcal{U} = \emptyset$ and $U = \Omega$.

Step 2 Let α and β denote, respectively, the lower and upper endpoints of U . Find $T(\alpha)$, a MST at α , with weight $W(\alpha)$.

Step 3 Let \bar{l} be as defined in (3.5).

Step 4 Put $q_l(W(\alpha)) = q_l(W(\alpha)) + P\{F\}$, where F is defined by (3.2).

Step 5 For $j \in \mathcal{A}$: If $\bar{l}_j \neq \beta_j$, file U_j in \mathcal{U} , where U_j is defined by (3.3).

Step 6 If $\mathcal{U} = \emptyset$, stop. Otherwise, remove a set U from \mathcal{U} and go to step 2.

Note that, in view of (3.5), at most $n - 1$ undetermined sets will be filed in step 5. Fewer than that number are filed if $\alpha_j = \beta_j$ for some $j \in T(\alpha)$. For that reason, if there are alternate optimal trees at α we would like to choose the one that contains as many tree arcs as possible with $\alpha_j = \beta_j$. In order to accomplish this, we preprocess the arcs in a manner similar to that used in STDA-II. Specifically, if

$\alpha_j = \beta_j$ we replace the arc weight $w_j(\alpha_j)$ by $w_j(\alpha_j) - \epsilon$, where ϵ is suitably small. After $T(\alpha)$ is found, the original arc weights are restored so that $W(\alpha)$ may be accurately computed.

As usual, STDIST may be terminated prior to complete partitioning of Ω to produce bounds. To produce bounds on $F(d) = P\{W(\mathbf{X}) \leq d\}$ for all d , we substitute step $\tilde{6}$ for step 6, and add steps 7 and 8.

Step $\tilde{6}$ If $\mathcal{U} = \emptyset$, stop. If a stopping condition is met, go to step 7. Otherwise, remove a set U from \mathcal{U} and go to step 2.

Step 7 For $d = W(1, \dots, 1), \dots, W(n_1, \dots, n_a)$: let $F_l(d) = F_u(d) = \sum_{i=W(1, \dots, 1)}^d q_l(i)$.

Step 8 For all $U \in \mathcal{U}$: Let α and β denote, respectively, the lower and upper endpoints of U . Find $W(\alpha)$ and $W(\beta)$.

For $W(\alpha) \leq d < W(\beta)$: Set $F_u(d) = F_u(d) + P\{U\}$.

For $W(\beta) \leq d \leq W(n_1, \dots, n_a)$: Set $F_u(d) = F_u(d) + P\{U\}$ and $F_l(d) = F_l(d) + P\{U\}$.

3.3 Computing Criticality Indices for the Stochastic MST Problem

We now describe a procedure for computing $P^F(e) = P\{\text{arc } e \text{ is on a MST}\}$, the criticality index for arc e as given in Definition 1.1. We first show that this problem is #P-hard.

Proposition 3.3 *The evaluation of $P^F(e) = P\{\text{arc } e \text{ is on a MST}\}$ is #P-hard.*

Proof: Let $G = (\mathcal{N}, \mathcal{A})$ be a graph whose arcs have random lengths with possible values in $\{0, 1\}$. Let $s, t \in \mathcal{N}$ and let \mathbf{X} be the vector of arc weights, which correspond here to lengths. Let $L(\mathbf{X})$ designate the (random) length of a shortest $s - t$ path in G when arc lengths are given by \mathbf{X} . The evaluation of $P\{L(\mathbf{X}) \geq 1\}$ is #P-hard [3].

Now create an arc $e = (s, t)$ with fixed length 1 and let $G' = (\mathcal{N}, \mathcal{A} + e)$. Then evaluating $P\{\text{arc } e \text{ is on a MST in } G'\}$ is equivalent to evaluating $P\{L(\mathbf{X}) \geq 1\}$. To see this, we need only establish that for a given realization x of arc lengths, e is on a MST in G' if and only if $L(x) \geq 1$. This is proved most easily by contrapositive.

Suppose e can be on no MST in G' . Consider a MST T , and let $C(e)$ be the unique cycle that e forms with T . Every arc in $C(e) - e$ must have zero length, for otherwise e could be pivoted in to replace an arc in $C(e) - e$ with length 1, yielding an MST containing e . But $C(e) - e$ necessarily is an $s - t$ path with zero length. Then $L(x) = 0$.

Now suppose $L(x) = 0$. Consider P , an $s - t$ path in G of length 0, and an arbitrary MST T' in G' . If $P' = P \setminus T'$ is nonempty, then we may pivot each arc in P' into T' one at a time, arriving at a new MST T containing P . Since T is acyclic, $e \notin T$. Furthermore, since necessarily $C(e) - e = P$, e cannot be pivoted in to achieve a MST containing e . Then no MST contains e .

Therefore,

$$P^F(e) = P\{\text{arc } e \text{ is on a MST}\} = P\{L(\mathbf{X}) \geq 1\}$$

and the evaluation of $P^F(e)$ is #P-hard. \square

In finding $P^F(e)$, the feasible region consists of all points in Ω at which arc e is on a MST. Before presenting any details of the GSD solution of this problem, though, we need to investigate the nature of feasibility in this setting.

Proposition 3.4 *Let $\bar{l} \in \Omega$, and let the weight of arc $j \in \mathcal{A}$ be $w_j(\bar{l}_j)$. Suppose that arc e is in a MST at this level. Then e is in a MST for all states in the set*

$$F = \{l \in U : 1 \leq l_e \leq \bar{l}_e \quad (3.6)$$

$$\bar{l}_j \leq l_j \leq n_j \quad \text{for } j \neq e\}.$$

Proof: All points in F can be arrived at from \bar{l} by a sequence of steps, each involving a single arc. Possible steps involve either decreasing the weight of e or increasing

that of an arc $j \neq e$ (either $j \in T$ or $j \notin T$). Thus we need only show that each such step preserves the requirement that e is on a MST. We consider each of three types of steps separately.

i) Increasing the weight of arc $j \neq e, j \in T$. By Property 4 (Section 3.1), this step either does not affect the MST T or causes the MST to change to $T - j + i$, where $i \notin T$. In either case, since $j \neq e$, e can stay on a MST.

ii) Increasing the weight of arc $j \neq e, j \notin T$. By Property 3, this step cannot force a change in T . Thus e is still on a MST.

iii) Decreasing the weight of arc $e \in T$. By Property 1, T is still a MST, and $e \in T$.

Therefore, if \bar{l} is feasible then F is a feasible set. \square

Proposition 3.5 *Let $\bar{l} \in \Omega$, and let the weight of arc $j \in \mathcal{A}$ be $w_j(\bar{l}_j)$. Suppose that arc e is in no MST at this level. Then e is in no MST for any point in the set*

$$I = \{l \in U : \bar{l}_e \leq l_e \leq n_e \quad (3.7)$$

$$1 \leq l_j \leq \bar{l}_j \quad \text{for } j \neq e\}.$$

Proof: All points in I can be arrived at from \bar{l} by a sequence of steps, each involving a single arc. Possible steps involve either increasing the weight of arc e or decreasing the weight of an arc $j \neq e$ (either $j \in T$ or $j \notin T$). Thus we need only show that each such step preserves the requirement that e can be on no MST. Note that this property is equivalent to requiring that the weight of arc e (strictly) exceeds the weights of all arcs in $C(e) - e$, where $C(e)$ is the unique cycle that e forms with the MST T . We consider each of three types of steps separately.

i) Decreasing the weight of arc $j \neq e, j \in T$. If $j \in C(e) - e$, then the weight of the most heavily weighted arc in $C(e) - e$ either stays the same or is decreased. If $j \notin C(e) - e$, then $C(e) - e$ is unchanged. In all of these cases, the weight of arc e still exceeds that of the most heavily weighted arc in $C(e) - e$, so that e is in no MST.

ii) Decreasing the weight of arc $j \neq e$, $j \notin T$. By Property 2, this step either does not impact T , in which case e clearly is still in no MST, or causes an arc $i \in T$ to be displaced from T by arc j . If $i \notin C(e) - e$, the cycle that e forms with the new MST is the same as it was before, namely $C(e) - e$, so e still cannot pivot in. If $i \in C(e) - e$, then the cycle that e forms with the new MST $T - i + j$ is different from $C(e) - e$. Specifically, the new cycle includes the remnants of the old cycle, $C(e) \setminus \{e, i\}$, plus the remnants of the cycle j formed with T , $C(j) - i$. The weight of e still exceeds that of all arcs in $C(e) \setminus \{e, i\}$. Furthermore, it must be that arc i was the most heavily weighted arc in $C(j) - j$ (since i was pivoted out). Since we know the weight of e exceeds that of arc i (it couldn't displace it), it also exceeds the weights of all arcs in $C(j)$. So e still cannot displace any arc in its cycle with the new MST, therefore it can be on no MST.

iii) Increasing the weight of arc $e \notin T$. Clearly, the weight of arc e still exceeds that of all arcs in $C(e) - e$, so that e can be on no MST.

Therefore, if \bar{l} is infeasible then I is an infeasible set. \square

An immediate consequence of these propositions is that the random variable X_e is lower feasible while the remaining variables are upper feasible. Thus for $U = [\alpha, \beta]$ the extreme feasible candidate is the point $(\alpha_e | \beta) \equiv (\beta_1, \dots, \beta_{e-1}, \alpha_e, \beta_{e+1}, \dots, \beta_n)$ and the extreme infeasible candidate is $(\beta_e | \alpha) \equiv (\alpha_1, \dots, \alpha_{e-1}, \beta_e, \alpha_{e+1}, \dots, \alpha_n)$. As with other GSD applications, the above propositions also imply that we may pre-screen an undetermined set $U = [\alpha, \beta]$ by evaluating the extremes. If $(\alpha_e | \beta)$ is infeasible then U is infeasible, and if $(\beta_e | \alpha)$ is feasible then U is feasible.

We shall now describe procedure STCRIT. As in STDIST, limiting indices \hat{l}_i will not be used. Thus decomposition from the feasible extreme, in which we push in from extreme feasible point to derive a feasible point \bar{l} , will yield one feasible set

$$F = \{l \in U : \alpha_e \leq l_e \leq \bar{l}_e \quad (3.8)$$

$$\bar{l}_i \leq l_i \leq \beta_i \text{ for } i \neq e\},$$

the undetermined set

$$U_e = \{l \in U : \bar{l}_e + 1 \leq l_e \leq \beta_e \quad (3.9)$$

$$\alpha_i \leq l_i \leq \beta_i \quad \text{for } i \neq e\} (= \emptyset \text{ if } \bar{l}_e = \beta_e),$$

and for $j \neq e$ the undetermined sets

$$U_j = \{l \in U : \quad \bar{l}_i \leq l_i \leq \beta_i \quad \text{for } i < j, i \neq e$$

$$\alpha_j \leq l_j \leq \bar{l}_j - 1 \quad (3.10)$$

$$\alpha_i \leq l_i \leq \beta_i \quad \text{for } i > j, i \neq e$$

$$\alpha_e \leq l_e \leq \bar{l}_e \quad \} (= \emptyset \text{ if } \bar{l}_j = \alpha_j).$$

Decomposition from the infeasible extreme, in which we push in from the extreme infeasible point to derive an infeasible point \bar{l} , will yield one infeasible set

$$I = \{l \in U : \alpha_e \leq l_e \leq \bar{l}_e \quad (3.11)$$

$$\alpha_i \leq l_i \leq \bar{l}_i \quad \text{for } i \neq e\},$$

the undetermined set

$$U_e = \{l \in U : \alpha_e \leq l_e \leq \bar{l}_e - 1 \quad (3.12)$$

$$\alpha_i \leq l_i \leq \beta_i \quad \text{for } i \neq e\} (= \emptyset \text{ if } \bar{l}_e = \alpha_e),$$

and for $j \neq e$ the undetermined sets

$$U_j = \{l \in U : \quad \alpha_i \leq l_i \leq \bar{l}_i \quad \text{for } i < j, i \neq e$$

$$\bar{l}_j + 1 \leq l_j \leq \beta_j \quad (3.13)$$

$$\alpha_i \leq l_i \leq \beta_i \quad \text{for } i > j, i \neq e$$

$$\bar{l}_e \leq l_e \leq \beta_e \quad \} (= \emptyset \text{ if } \bar{l}_j = \beta_j).$$

As in STDA-II, the prescreening phase and the phase in which we decide whether to decompose feasibly or infeasibly are somewhat free-flowing. A number of possible feasible and infeasible points are investigated first, in hope of deriving a particularly

effective and/or easily obtained cutoff state. If that fails, we may choose between default feasible and infeasible decompositions. Throughout, Properties 1 through 4 from Section 3.1 will be used. Steps 11 through 14 may be used if bounds on $P^F(e)$ are desired. We present this procedure in narrative form.

Procedure STCRIT(e)

Step 1 Start with $U = \Omega$. Set $\mathcal{U} = \emptyset$ and $P_l^F(e) = 0$.

Step 2 Let α and β denote, respectively, the lower and upper endpoints of U . Find $T(\beta_e|\alpha)$, a MST at the extreme infeasible candidate. If $e \in T(\beta_e|\alpha)$, go to step 3. Otherwise, find i_1 , the most heavily weighted arc on the cycle that arc e forms with $T(\beta_e|\alpha)$. If arc e and arc i_1 have equal weight, pivot arc e into $T(\beta_e|\alpha)$.

Step 3 If $e \notin T(\beta_e|\alpha)$, go to step 4. Otherwise, $(\beta_e|\alpha)$ is a feasible point so that U is feasible. Go to step 11.

Step 4 If $w_e(\alpha_e) > w_{i_1}(\alpha_{i_1})$ then $T(\beta_e|\alpha)$ is a MST at levels α (call it $T(\alpha)$); go to step 5. Otherwise, decreasing the weight of arc e will cause it to displace arc i_1 . Let $\bar{l}_e = \max\{\gamma : \alpha_e \leq \gamma \leq \beta_e; w_e(\gamma) \leq w_{i_1}(\alpha_{i_1})\}$ and $\bar{l}_j = \alpha_j$ for $j \neq e$. Perform feasible decomposition and go to step 11.

Step 5 Find $T(\alpha_e|\beta)$, a MST at the extreme feasible candidate. If $e \in T(\alpha_e|\beta)$, go to step 6. Otherwise, find i_2 , the most heavily weighted arc on the cycle that e forms with $T(\alpha_e|\beta)$. If arc e and arc i_2 have equal weight, pivot arc e into $T(\alpha_e|\beta)$ and go to step 6. Otherwise, e is on no MST at $(\alpha_e|\beta)$, which is thus an infeasible point. Since U is infeasible, go to step 11.

Step 6 We have $e \in T(\alpha_e|\beta)$. We now seek to force e out of the MST by changing the levels of only a few arcs, yielding an infeasible point. One preprocessing pass through the arcs not in $T(\alpha_e|\beta)$ (similar to the procedure SHORT in Section 3.2.1, but slightly more involved) finds the following:

- j_1 = the least weighted arc off $T(\alpha_e|\beta)$ whose cycle contains arc e , which could displace arc e if the weight of e is increased (in step 7).
- j_2 = an arc not in $T(\alpha_e|\beta)$ which has e (at level β) as its most heavily weighted cycle arc and for which $w_{j_2}(\alpha_{j_2}) < w_e(\beta_e)$. If the weight of e has been increased to $w_e(\beta_e)$ (in step 7), then j_2 can be lowered (in step 8) to force e out of the tree. If no such arc exists, let $j_2 = 0$.
- j_3 = an arc not in $T(\alpha_e|\beta)$ whose cycle with $T(\alpha_e|\beta)$ contains e (not as the most heavily weighted cycle arc), for which $w_j(\alpha_j) < w_e(\beta_e)$ for all $j \in C(j_3) - e$ (for use in step 9). If no such arc is found, set $j_3 = 0$.

Note: Even though all these arcs might not be needed, we may efficiently determine all three in one $O(an)$ procedure, essentially the same amount of work needed to determine any one of these numbers.

Step 7 If $w_e(\beta_e) \leq w_{j_1}(\beta_{j_1})$, then increasing the weight of arc e to $w_e(\beta_e)$ does not force it out of the current MST; go to step 8. Otherwise, this increase causes arc e to be displaced by j_1 . Set $\bar{l}_e = \min\{\gamma : \alpha_e \leq \gamma \leq \beta_e : w_e(\gamma) > w_{j_1}(\beta_{j_1})\}$ and $\bar{l}_j = \beta_j$ for $j \neq e$. Perform infeasible decomposition and go to step 11.

Step 8 $T(\alpha_e|\beta)$ is now a MST at levels β ; call it $T(\beta)$. If $j_2 = 0$, go to step 9. Otherwise, decreasing the weight of j_2 will cause it to displace e from $T(\beta)$. Set $\bar{l}_e = \beta_e$, $\bar{l}_{j_2} = \max\{\gamma : \alpha_{j_2} \leq \gamma \leq \beta_{j_2}; w_{j_2}(\gamma) < w_e(\beta_e)\}$ and $\bar{l}_j = \beta_j$ for all other arcs j . Perform infeasible decomposition and go to step 11.

Step 9 If $j_3 = 0$, go to step 10. Otherwise, for $j \in C(j_3) - j_3 - e$, set $\bar{l}_j = \max\{\gamma : \alpha_j \leq \gamma \leq \beta_j; w_j(\gamma) < w_e(\beta_e)\}$ and $\bar{l}_e = \beta_e$. Then e is the most heavily weighted arc on the cycle. Set $\bar{l}_{j_3} = \max\{\gamma : \alpha_{j_3} \leq \gamma \leq \beta_{j_3}; w_{j_3}(\gamma) < w_e(\beta_e)\}$ to displace e . Set $\bar{l}_j = \beta_j$ for all arcs j whose weights are not yet set. Perform infeasible decomposition and go to step 11.

Step 10 We are left to choose from the default decompositions. Do one of the following:

- Derive a feasible point from the known feasible point β by setting $\bar{l}_j = \alpha_j$ for $j \in T(\beta)$, $\bar{l}_e = \beta_e$, and $\bar{l}_j = \min\{\gamma : \alpha_j \leq \gamma \leq \beta_j; j \text{ does not displace } e \text{ from } T(\beta)\}$ for $j \notin T(\beta)$. Perform feasible decomposition and go to step 11.
- Derive an infeasible point from the known infeasible point α (which we know to be infeasible from step 4 above) by setting $\bar{l}_j = \beta_j$ for $j \notin T(\alpha)$, $\bar{l}_e = \alpha_e$, and $\bar{l}_j = \max\{\gamma : \alpha_j \leq \gamma \leq \beta_j; j \text{ stays in } T(\alpha) \text{ when arcs } j \notin T(\alpha) \text{ are at levels } \alpha_j\}$ for $j \in T(\alpha)$. Perform infeasible decomposition and go to step 11.

Step 11 If one or more feasible intervals have been generated, add their probability to $P_l^F(e)$. If $\mathcal{U} = \emptyset$, stop. If a stopping condition is met, go to step 12. Otherwise, remove a set U from \mathcal{U} and go to step 2.

Step 12 Set $P_u^F(e) = P_l^F(e)$

Step 13 For all $U \in \mathcal{U}$: Let α and β denote, respectively, the lower and upper limiting points of U . Find $T(\alpha_e|\beta)$. If $e \in T(\alpha_e|\beta)$ or if e can be pivoted into $T(\alpha_e|\beta)$ yielding an alternate MST, set $P_u^F(e) = P_u^F(e) + P\{U\}$.

Step 14 Return bounds $P_l(e) \leq P^F(e) \leq P_u(e)$.

A few comments are in order concerning STCRIT. In steps 3 through 9 several attempts are made to find feasible or infeasible points that are easy to find and that will yield few undetermined sets. If U is found to be all feasible in step 3, clearly no undetermined sets are produced. If reducing arc e in step 4 yields a feasible point, then one undetermined set will be produced. Finding U to be infeasible in step 5 produces no undetermined sets. If raising arc e from $(\alpha_e|\beta)$ causes it to pivot out of the tree in step 7, then one undetermined set is produced. If arc j_2 can be lowered so as to displace arc e from $T(\beta)$ in step 8, two undetermined sets are produced. Finally, if step 9 succeeds in identifying an infeasible set we get undetermined sets

for arc e , for j_3 and for each of the arcs on the cycle formed by j_3 with $T(\beta)$. If we are left with the default decompositions of step 10, we get up to $n - 1$ undetermined sets (if we decompose feasibly) or up to $a - n + 1$ (if we decompose infeasibly). In that case, the decomposition yielding fewest new undetermined sets is used.

One final comment about STCRIT is that it can be used with little modification in a way similar to that given in [1] to compute the $P\{\text{arc } e \text{ is on a MST and } W(\mathbf{X}) > d\}$. Combining this measure with the output of STDA-I or STDA-II, we can compute the *conditional* criticality index

$$P\{\text{arc } e \text{ is on a MST} \mid W(\mathbf{X}) > d\} = \frac{P\{\text{arc } e \text{ is on a MST and } W(\mathbf{X}) > d\}}{1 - P\{W(\mathbf{X}) \leq d\}}.$$

These probabilities are useful in that they identify the arcs that are likely to be MST members when the network cannot be connected feasibly. Unfortunately, it takes a bit of extra work to produce them. We first run STDA-I or STDA-II, filing all generated infeasible sets in a list \mathcal{I} . If these are run to complete decomposition, the union of the sets in \mathcal{I} is precisely the infeasible region of Ω . These infeasible sets are passed as input to STCRIT (rather than starting with $\mathcal{U} = \emptyset$ in step 1, we begin with $\mathcal{U} = \mathcal{I}$ and instead of using Ω as the first undetermined set we simply choose U from \mathcal{U}).

In conclusion, STCRIT is an algorithm which can produce marginal, joint or conditional criticality indices for any arc $e \in \mathcal{A}$. These measures are useful in determining which arcs are influential in the formation of minimum spanning trees. Section 3.4 below contains examples.

3.4 Stochastic MST Examples

In this section we demonstrate the performance of the various routines developed in the present chapter. For each example, we execute STDA-I (Pure Strategies 1 and 2, Mixed Strategies 1 through 3), STDA-II (Pure Strategy 1 only), STDIST and STCRIT. Our main goals are to compare the performance of several versions

of STDA-I and STDA-II, and to show examples of complete decomposition as compared to partial decomposition to produce bounds. All codes were implemented in FORTRAN 77 and were compiled and executed on a SUN SPARCstation IPC. The implementation of Kruskal's algorithm in Camerini, Galbiati and Maffioli [12] was used for computing a minimum spanning tree. In all applications, the list of undetermined sets was kept in a heap sorted by probability until the most probable undetermined set had probability less than 10^{-20} . Arc weight distributions represent what we feel to be reasonable situations. Specifically, a high probability is assigned to the lowest weight (the *status quo*) and low probabilities are assigned to the higher weights (the occasional deviations from the *status quo*). All CPU times quoted are in seconds. In all tables, $P\{\mathcal{U}\}$ represents the sum of the probabilities of undetermined sets contained in the collection \mathcal{U} .

Example 3.1 (Example 1.1 revisited) The network in Table 3.1 has 10 nodes and 21 arcs. Each arc has three possible weights. The state space has more than 10 billion states. For this example, the range of possible MST weights is from $W(1, \dots, 1) = 47$ to $W(3, \dots, 3) = 138$. We use STDA-I, STDA-II and ST-HYBRID to compute $P\{W(\mathbf{X}) \leq 60\}$ and $P\{W(\mathbf{X}) \leq 90\}$, these tree weights being relatively extreme possible values, chosen to demonstrate the difference in feasible and infeasible decompositions. The results for $F(60)$ are in Table 3.2, for $F(90)$ in Table 3.3. A partial listing of the cumulative distribution function $F(d)$ (from STDIST) is given in Table 3.4. Full decomposition used 445.91 CPU seconds and decomposed 634,405 sets. Partial decomposition decomposed 100,000 intervals in 69.63 CPU seconds. Criticality indices (from STCRIT) are in Table 3.5. Arc 10, which stands out as requiring more computational effort than other arcs, achieved bounds of 0.0001852 and 0.0003542 in 6.81 seconds after decomposing 500 intervals.

There are a few items worthy of comment in the GSD solutions of Example 3.1. First, we note the incredible efficiency of the procedure STCRIT. Note that almost all arcs required the evaluation of fewer than 200 undetermined intervals. Since each such evaluation entails two MST evaluations, most arcs required the evaluation of

Table 3.1: Input Distribution for Example 3.1.

Arc	Weight (Probability)		
1 = (1,2)	2.0 (0.90)	8.0 (0.08)	12.0 (0.02)
2 = (1,3)	10.0 (0.85)	24.0 (0.12)	35.0 (0.03)
3 = (1,4)	6.0 (0.88)	18.0 (0.10)	24.0 (0.02)
4 = (2,5)	17.0 (0.75)	35.0 (0.20)	50.0 (0.05)
5 = (2,6)	3.0 (0.68)	7.0 (0.25)	10.0 (0.07)
6 = (3,2)	12.0 (0.85)	22.0 (0.11)	30.0 (0.04)
7 = (3,7)	10.0 (0.80)	19.0 (0.14)	24.0 (0.06)
8 = (3,8)	4.0 (0.75)	6.0 (0.17)	10.0 (0.08)
9 = (4,3)	3.0 (0.84)	7.0 (0.13)	12.0 (0.03)
10 = (4,9)	21.0 (0.85)	33.0 (0.09)	45.0 (0.06)
11 = (5,7)	8.0 (0.77)	14.0 (0.13)	18.0 (0.10)
12 = (5,10)	15.0 (0.76)	21.0 (0.22)	25.0 (0.02)
13 = (6,3)	5.0 (0.65)	10.0 (0.23)	12.0 (0.12)
14 = (6,5)	18.0 (0.94)	27.0 (0.05)	36.0 (0.01)
15 = (6,7)	25.0 (0.80)	38.0 (0.12)	50.0 (0.08)
16 = (7,8)	20.0 (0.87)	30.0 (0.09)	40.0 (0.04)
17 = (7,10)	4.0 (0.75)	9.0 (0.14)	15.0 (0.11)
18 = (8,4)	9.0 (0.82)	13.0 (0.15)	20.0 (0.03)
19 = (8,9)	15.0 (0.79)	28.0 (0.15)	45.0 (0.06)
20 = (9,7)	10.0 (0.95)	17.0 (0.03)	30.0 (0.02)
21 = (9,10)	8.0 (0.85)	14.0 (0.10)	25.0 (0.05)

about 400 states, or a fraction 3.8×10^{-8} of the state space. Even arc 10, whose criticality index required the evaluation of many more states, only looked at the fraction 8.0×10^{-6} of the state space.

Secondly, we comment on the bounds produced by STDIST for Table 3.4. Clearly, the bounds corresponding to small MST lengths (small values of d) converge more quickly than those for large values of d . This is due to the manner in which these probabilities are accumulated, starting at the feasible extreme and moving up.

Example 3.2 We consider again the network in Example 3.1, but with revised probability structure. Here, each arc has two possible weights, as shown in Table 3.6.

Table 3.2: Computing $F(60) = 0.8267495828$ for Example 3.1.

Routine	Full Decomposition		500 Sets Decomposed				
	Sets	Time	Bounds		Time	$ \mathcal{U} $	$P\{\mathcal{U}\}$
STDA-I P1	790	5.63	.8197061589	.8278633467	3.67	192	.008157
STDA-I P2	7886	68.31	.6655013736	.9071519952	4.67	620	.2417
STDA-I M1	648	5.86	.8231632665	.8273711991	4.61	123	.004208
STDA-I M2	5582	48.56	.6614555582	.8989402274	4.97	585	.2375
STDA-I M3	4822	40.41	.7109925528	.8718820125	5.02	689	.1609
STDA-II P1	1421	1.30	.8188268475	.8298433389	0.67	128	.01102
ST-HYBRID	2660	10.24	.8065717649	.8416536725	3.01	324	.03508

Table 3.3: Computing $F(90) = 0.9999950999$ for Example 3.1.

Routine	Full Decomposition		500 Sets Decomposed				
	Sets	Time	Bounds		Time	$ \mathcal{U} $	$P\{\mathcal{U}\}$
STDA-I P1	247853	1603.52	.94844450	1.0000000	3.39	516	.05156
STDA-I P2	205576	1655.32	.98419966	.99999965	5.09	2506	.01580
STDA-I M1	88857	727.00	.98871880	.99999990	4.89	1315	.01128
STDA-I M2	171400	1389.58	.96960607	.99999991	4.53	647	.03039
STDA-I M3	195337	1589.95	.98216012	.99999966	5.54	2329	.01784
STDA-II P1	127683	95.18	.99899010	.99999999	0.76	686	.001010
ST-HYBRID	506168	2115.43	.90042352	.99999991	2.37	573	.09958

Table 3.4: Partial listing of the cdf $F(d)$ for Example 3.1.

Full Decomposition		100,000 Sets Decomposed	
d	$F(d)$	Lower	Upper
47.0	0.0984162	0.0984161	0.0984161
48.0	0.1450503	0.1450503	0.1450503
49.0	0.1838571	0.1838571	0.1838571
50.0	0.2594979	0.2594979	0.2594979
51.0	0.2790302	0.2790302	0.2790302
52.0	0.3801882	0.3801882	0.3801882
53.0	0.4426021	0.4426021	0.4426021
54.0	0.5169705	0.5169705	0.5169705
55.0	0.5817478	0.5817478	0.5817478
56.0	0.6376404	0.6376404	0.6376404
57.0	0.6932358	0.6931849	0.6932502
58.0	0.7424916	0.7423291	0.7425250
59.0	0.7888656	0.7885259	0.7889317
60.0	0.8267496	0.8260885	0.8268742
61.0	0.8621487	0.8609763	0.8623779
62.0	0.8894412	0.8873433	0.8898221
63.0	0.9126231	0.9093929	0.9132196
64.0	0.9316627	0.9270028	0.9325470
65.0	0.9468859	0.9402229	0.9481127
70.0	0.9874525	0.9684075	0.9909956
75.0	0.9976452	0.9736052	0.9994587
80.0	0.9996402	0.9782001	0.9999404
85.0	0.9999545	0.9834824	0.9999924
90.0	0.9999951	0.9870540	0.9999992
100.0	0.9999999	0.9945591	0.9999999

Table 3.5: Criticality indices for Example 3.1.

Arc	Criticality Index	# Sets	Time
1	0.9393247	12	0.071
2	0.0432205	11	0.065
3	0.5859462	6	0.036
4	0.0490331	3199	18.94
5	0.8227428	200	1.18
6	0.0021765	8	0.047
7	0.8004465	77	0.46
8	0.9353420	198	1.17
9	0.9084087	9	0.053
10	0.0001994	41939	248.28
11	0.9067588	174	1.03
12	0.0833359	9	0.053
13	0.6990894	16	0.095
14	0.0099071	169	1.00
15	0.0000000	1	0.006
16	0.0001650	12	0.071
17	0.9007350	4	0.024
18	0.0764335	10	0.059
19	0.1628822	13	0.077
20	0.2318770	10	0.059
21	0.8654815	8	0.047

Table 3.6: Input Distribution for Example 3.2.

Arc	Weight (Probability)	
1 = (1,2)	70.0 (0.95)	94.0 (0.05)
2 = (1,3)	25.0 (0.80)	52.0 (0.20)
3 = (1,4)	42.0 (0.70)	61.0 (0.30)
4 = (2,5)	26.0 (0.85)	50.0 (0.15)
5 = (2,6)	58.0 (0.70)	95.0 (0.30)
6 = (3,2)	15.0 (0.80)	43.0 (0.20)
7 = (3,7)	65.0 (0.60)	75.0 (0.40)
8 = (3,8)	59.0 (0.70)	98.0 (0.30)
9 = (4,3)	21.0 (0.90)	68.0 (0.10)
10 = (4,9)	89.0 (0.85)	96.0 (0.15)
11 = (5,7)	32.0 (0.80)	67.0 (0.20)
12 = (5,10)	63.0 (0.75)	99.0 (0.25)
13 = (6,3)	66.0 (0.80)	98.0 (0.20)
14 = (6,5)	60.0 (0.90)	88.0 (0.10)
15 = (6,7)	32.0 (0.75)	48.0 (0.25)
16 = (7,8)	16.0 (0.7)	42.0 (0.30)
17 = (7,10)	56.0 (0.80)	71.0 (0.20)
18 = (8,4)	90.0 (0.95)	96.0 (0.05)
19 = (8,9)	17.0 (0.80)	45.0 (0.20)
20 = (9,7)	3.0 (0.60)	15.0 (0.40)
21 = (9,10)	50.0 (0.75)	66.0 (0.25)

For this example, the possible MST weights range from 220 to 444. The cardinality of the state space is just over 2 million. In Table 3.7, we use STDA-I, STDA-II and ST-HYBRID to compute $F(280)$, and in Table 3.8 we give results for $F(400)$. A partial listing of the cdf $F(d)$ is given in Table 3.9. Full decomposition required 6.96 CPU seconds to decompose 9,743 intervals. Table 3.10 contains criticality indices.

Again, we see the efficiency of STCRIT, which used just 539 interval evaluations *total* to compute all 21 criticality indices. Most arcs required the evaluation of fewer than 20 states, a fraction 9.5×10^{-6} of the state space.

Table 3.7: Computing $F(280) = 0.8567535213$ for Example 3.2.

Routine	Full Decomposition	
	Sets	Time
STDA-I Pure 1	740	3.93
STDA-I Pure 2	356	2.39
STDA-I Mixed 1	289	2.46
STDA-I Mixed 2	336	3.03
STDA-I Mixed 3	295	2.63
STDA-II Pure 1	658	0.62
ST-HYBRID	573	3.20

Table 3.8: Computing $F(400) = 0.9999977607$ for Example 3.2.

Routine	Full Decomposition	
	Sets	Time
STDA-I Pure 1	740	3.93
STDA-I Pure 2	78	0.76
STDA-I Mixed 1	176	1.39
STDA-I Mixed 2	143	1.16
STDA-I Mixed 3	109	0.99
STDA-II Pure 1	465	0.46
ST-HYBRID	462	2.20

Table 3.9: Partial listing of the cdf $F(d)$ for Example 3.2.

d	$F(d)$	d	$F(d)$
220.0	0.0925345	275.0	0.8182486
221.0	0.1242605		
		280.0	0.8567536
225.0	0.1242605		
226.0	0.1489364	285.0	0.8871976
227.0	0.1573967		
		290.0	0.9202321
231.0	0.1573967		
232.0	0.2190863	300.0	0.9554863
233.0	0.2448637		
234.0	0.2464500	310.0	0.9774270
235.0	0.2464500		
236.0	0.2788370	320.0	0.9892763
237.0	0.3061347		
238.0	0.3281373	330.0	0.9953135
239.0	0.3337775		
240.0	0.3337775	340.0	0.9979105
241.0	0.3409746		
242.0	0.3516675	350.0	0.9991380
243.0	0.3588058		
244.0	0.3766160	360.0	0.9997070
245.0	0.3852992		
246.0	0.4104818	370.0	0.9998960
247.0	0.4297512		
248.0	0.4810531	380.0	0.9999668
249.0	0.5093677		
250.0	0.5187621	390.0	0.9999884
255.0	0.5673748	400.0	0.9999978
260.0	0.6490759	410.0	0.9999997
265.0	0.7218336	420.0	0.9999999
270.0	0.7599369	440.0	1.0000000

Table 3.10: Criticality indices for Example 3.2.

Arc	Criticality Index	# Sets	Time
1	0.0000000	1	0.0066
2	0.8740000	4	0.0263
3	0.2260000	5	0.0329
4	1.0000000	1	0.0066
5	0.1400000	3	0.0197
6	1.0000000	1	0.0066
7	0.0003105	33	0.2168
8	0.0420000	4	0.0263
9	0.9000000	2	0.0131
10	0.0000000	1	0.0066
11	0.8000360	8	0.0526
12	0.0387825	74	0.4862
13	0.0001656	46	0.3022
14	0.0162000	5	0.0329
15	1.0000000	1	0.0066
16	0.7600000	3	0.0197
17	0.2000000	3	0.0197
18	0.0000000	1	0.0066
19	0.2400000	3	0.0197
20	1.0000000	1	0.0066
21	0.7625270	339	2.2272

3.5 Extension of MST Results to Other Stochastic Matroid Settings

The GSD applications given in this chapter were successful due in no small measure to the matroid structure of the MST problem. Note that spanning trees are bases for the graphic matroid with ground set \mathcal{A} and *independent sets* being acyclic subgraphs. The following matroid properties were instrumental in making these applications more effective:

- Matroid problems can be solved by the greedy algorithm (here, Kruskal's method), so that evaluating a state point (finding an optimal matroid base for a given weighting of the elements of the ground set) can be done efficiently.
- When one element of the ground set changes weight, it is relatively easy to determine whether a given base is still optimal. Furthermore, by the exchangeability property of matroids it is easy to form a new optimal base by pivoting, assuming some means exist for finding the replacement element. This is important because the alternative is to solve a new optimization problem every time an element changes weight.

Since all matroid problems share these features to an extent, it is reasonable to assume that there are other stochastic matroid problems that will admit efficient solution by GSD. The second point is especially important. In our MST work, every exchange of elements was accomplished by investigating the unique cycle (minimal dependent set) formed by an arc not on a MST (an element of the ground set not in the optimal base) with the MST (the elements making up the optimal base). These comparisons resulted in efficient, unequivocal determination of the elements to be exchanged. All matroid problems share this feature to some degree. In the general (non-matroid) case, it is not always easy to determine whether weight changes cause a solution to cease being optimal, or to remedy the situation by identifying the elements to be exchanged. In the absence of some means for making these determi-

natons, every change in element weight will necessitate finding an optimal solution from scratch.

3.6 Conclusions

The GSD routines presented in this chapter represent real gains in our ability to characterize the behavior of minimum spanning trees in stochastic networks. They were particularly effective due to the attractive properties of matroids. We close this chapter with some observations about the computational results of Section 3.4 and some directions for further study.

It seems clear from Tables 3.2, 3.3, 3.7 and 3.8 that, despite its sophistication, STDA-I cannot compete with STDA-II in terms of speed. Based on our experience, it seems to be the case with GSD applications that frequently a simpler, clever decomposition scheme performs better than one that labors to cut off as much of an interval as possible in each iteration. In particular, even though the mixed strategies of STDA-I improved upon the STDA-I pure strategies, they do not compare favorably with STDA-II on the sample problems. It is possible that mixed strategies will work well with other GSD applications. It also seems clear that the hybrid routine did not outperform the best of the GSD routines either in terms of computational effort or in terms of achieving tight bounds.

Secondly, it is clearly beneficial in STDIST to accumulate probabilities for all possible MST weights at the same time. To see this, note that in Example 3.1 the fastest routine needed 95 seconds to compute $F(90)$. STDIST computed the entire distribution in an average of 4.85 seconds per possible MST weight.

Finally, even among MST routines, the procedure STCRIT stands out as a very efficient routine. It would be good, then, to find applications for this effective procedure. For example, a reasonable heuristic solution to the problem of identifying the most probable minimum spanning tree is suggested. After identifying criticality indices for all arcs, we build a maximally acyclic connected graph by considering

arcs one at a time in decreasing order of their criticality index, discarding any arc that would form a cycle. Investigating the effectiveness of this proposed heuristic seems worthy of future research.

CHAPTER 4

The Stochastic Minimum Cost Network Flow Problem

4.1 Introduction

An important problem in the area of network optimization is that of satisfying demand for a commodity by using a least-cost subset of transportation arcs, with flow constrained on each arc by a finite upper bound. This problem is known as the (capacitated) minimum cost flow (MCF) problem.

In this chapter we consider flow networks in which neither the cost of using an arc nor the capacity of that arc is known with certainty. Rather, these values are given as discrete random variables. After some formal preliminaries, we shall present applications of the General State Space Decomposition (GSD) algorithm of Chapter 2 designed to assist in characterizing the probabilistic behavior of this system.

Consider a network $G = (\mathcal{N}, \mathcal{A}, s, t)$ with node set $\mathcal{N} = \{1, \dots, n\}$ and arc set $\mathcal{A} = \{1, \dots, a\}$. From \mathcal{N} , we designate nodes s and t as the unique *source* (supply of the commodity) and *sink* (demand for the commodity), respectively. Of course, this is not restrictive since the existence of multiple supply and/or demand locations can be modeled by creating a fictitious *supersource* s and a fictitious *supersink* t . We assume that supply at s equals demand at t , both given by $v \geq 0$. We assume that nodes do not restrict flow and that flow passes through nodes without incurring a cost. Nodes for which these assumptions do not hold may be split into two nodes connected by an arc with the desired cost and capacity properties.

Associated with each arc $j \in \mathcal{A}$ is a cost X_j per unit of flow transmitted, a discrete random variable of the form given in Chapter 2. That is, the cost of using arc j takes on finite values $c_j(1) \leq \dots \leq c_j(m_j)$ with respective probabilities $p_j(1), \dots, p_j(m_j)$. Furthermore, the random capacity (upper bound) of arc j , given by Y_j , takes on nonnegative finite values $u_j(1) \leq \dots \leq u_j(n_j)$ with probabilities $q_j(1), \dots, q_j(n_j)$. We represent the collection of cost and capacity random variables in vector form as $(\mathbf{X}|\mathbf{Y}) = (X_1, \dots, X_a | Y_1, \dots, Y_a)$. The state space Ω of \mathbf{X} contains $\prod_{j=1}^a m_j \prod_{j=1}^a n_j$ points of the form $(x|y) = (c_1(k_1), \dots, c_a(k_a) | u_1(l_1), \dots, u_a(l_a))$, where $k_j \in \{1, \dots, m_j\}$ and $l_j \in \{1, \dots, n_j\}$ for all $j \in \mathcal{A}$. As in Chapter 2, we shall use the convention of writing the state point $(x|y)$ as $(k|l) = (k_1, \dots, k_a | l_1, \dots, l_a)$. For a fixed $(x|y) \in \Omega$, let $C(x|y)$ be the cost of a MCF (with value v).

We consider first the following problem related to the stochastic MCF system described above.

Problem MC1 Let $d \geq 0$. Assuming the components of $(\mathbf{X}|\mathbf{Y})$ are independent, evaluate the probability $P\{C(\mathbf{X}|\mathbf{Y}) \leq d\}$ that there is a flow of v units whose cost is at most d .

For a given d , a realization $(k|l)$ is considered to be a feasible state if $C(k|l) \leq d$. Clearly, feasible points correspond to low arc costs and high arc capacities. Thus the cost variables X_j are lower feasible and the capacity variables Y_j are upper feasible. In Section 4.2 we compute $P\{C(\mathbf{X}|\mathbf{Y}) \leq d\}$ and give a GSD modification that simultaneously determines the entire probability distribution of $C(\mathbf{X}|\mathbf{Y})$.

In Problem MC2 below, we consider the case in which arc costs and capacities are dependent. The random vectors $Z_j = (X_j, Y_j)$ take on values $(c_j(1), u_j(1)), \dots, (c_j(n_j), u_j(n_j))$ with respective probabilities $p_j(1), \dots, p_j(n_j)$. We further assume that $u_j(1) \geq \dots \geq u_j(n_j)$ for all j . Let $\mathbf{Z} = (Z_1, \dots, Z_a)$. For a fixed realization z of \mathbf{Z} , let $C(z)$ be the cost of a MCF.

Problem MC2 Let $d \geq 0$. Assuming the components of \mathbf{Z} are independent, evaluate the probability $P\{C(\mathbf{Z}) \leq d\}$ that there is a flow of f units whose cost

does not exceed d .

Here again, feasible states are those having MCF cost no greater than d . Because of the reverse ordering of the capacity variables, we have low costs paired with high capacities so that Z_j is lower feasible for all j . We will address this problem in Section 4.3

The following result shows that these problems are #P-hard. It is stated for problem MC1, but clearly implies that problem MC2 is also #P-hard.

Proposition 4.1 *The evaluation of $P\{C(\mathbf{X}|\mathbf{Y}) \leq d\}$ is #P-hard.*

Proof: The evaluation of $P\{\text{maximum } s-t \text{ flow} \geq d\}$ is #P-hard for arbitrary $d \geq 0$ [3]. Consider an instance of this problem. Put in a return arc (t, s) with infinite capacity. Let the costs of all original arcs be 0 and the cost of arc (t, s) be -1 . Clearly, a minimum cost flow in the resulting network will ship as much as possible on arc (t, s) , and will thus yield a maximum $s-t$ flow in the original network. In fact, the events $\{C(\mathbf{X}|\mathbf{Y}) \leq -d\}$ and $\{\text{maximum } s-t \text{ flow} \geq d\}$ are equal.

Since the evaluation of $P\{\text{maximum } s-t \text{ flow} \geq d\}$ is #P-hard, so must be the evaluation of $P\{C(\mathbf{X}|\mathbf{Y}) \leq d\}$. \square

In Section 4.2 we shall present eight different GSD algorithms for solving problem MC1, one "hybrid" partitioning routine, and a GSD modification which simultaneously computes the entire distribution $C(\mathbf{X}|\mathbf{Y})$. Section 4.3 contains analogous routines applied to problem MC2 and the evaluation of $P\{C(\mathbf{Z}) \leq d\}$. Numerical examples are found in Section 4.4 and Section 4.5 offers concluding remarks.

4.2 The Case of Independent Costs and Capacities

We consider here the flow network whose arc costs X_j and arc capacities Y_j are mutually independent random variables. The system operates feasibly if demand is

satisfied and the constraint $C(\mathbf{X}|\mathbf{Y}) \leq d$ is satisfied. We wish to compute $P^F = P\{C(\mathbf{X}|\mathbf{Y}) \leq d\}$.

In the following subsections, we propose eleven algorithms for solving problem MC1. We begin in Sections 4.2.1 and 4.2.2 by describing two different approaches to deriving feasible and infeasible states. These operations, of course, are the heart of any partitioning routine. Based on each of these approaches, we present in Section 4.2.3 two GSD feasible decompositions and two GSD infeasible decompositions, and in Section 4.2.4 one hybrid routine (in which each interval is partitioned into one arbitrary feasible interval, one arbitrary infeasible interval and up to $4a - 1$ undetermined intervals). In Section 4.2.5 we describe the simultaneous evaluation of the entire distribution of $C(\mathbf{X}|\mathbf{Y})$.

4.2.1 Decomposition Algorithms I

In this section we describe techniques for deriving feasible and infeasible cutoff states for use in partitioning algorithms for evaluating P^F . These algorithms, which we collectively call the Minimum Cost Flow Decomposition Algorithms I (MCDA-I), have the common feature that feasible sets are removed by finding a feasible state $(\bar{k}|\bar{l})^F$ derived from the extreme feasible point $(\alpha|\beta)$ and infeasible sets are removed by deriving an infeasible state $(\bar{k}|\bar{l})^I$ from $(\beta|\alpha)$. We describe here the mechanics for deriving $(\bar{k}|\bar{l})^F$ and $(\bar{k}|\bar{l})^I$ for an undetermined interval $U = [\alpha, \beta]$ for which $C(\alpha|\beta) \leq d$ and $C(\beta|\alpha) > d$. Note that α and β are of the form $\alpha = (\alpha_1^c, \dots, \alpha_a^c | \alpha_1^u, \dots, \alpha_a^u)$ and $\beta = (\beta_1^c, \dots, \beta_a^c | \beta_1^u, \dots, \beta_a^u)$.

MCDA-I Feasible State Derivation

Here we seek to derive a feasible state $(\bar{k}|\bar{l})^F$, starting with $(\alpha|\beta)$, that will cut off as large a feasible set as is practical using Proposition 2.1.

We begin by finding a minimum cost feasible flow f° at $(\alpha|\beta)$. Clearly if $f^\circ(j) = 0$ for some $j \in \mathcal{A}$, we may raise the cost of arc j to $c_j(\beta_j)$ and reduce the capacity of arc j to $u_j(\alpha_j)$ without impacting optimality. Also, if $0 < f^\circ(j) < u_j(\beta_j)$ we may

reduce the capacity of arc j as long as it does not fall below $f^o(j)$.

Having made these simple changes, we are left to decide whether or not to attempt further modifications to $(\bar{k}|\bar{l})^F$. Of course, we are free to use $(\bar{k}|\bar{l})^F$ as it now stands since by Proposition 2.1 any feasible point will yield a feasible set. However, it is likely that most arcs carry *some* flow, so that in some cases our previous changes to $(\bar{k}|\bar{l})^F$ accomplish little. Thus we are tempted to try further cost increases and/or capacity reductions. At issue is the fact that such attempts must now come at the expense of re-optimizing. However, the existence of efficient codes for performing sensitivity analysis on network flow problems eases our concerns.

Since capacities have already been addressed to a degree, we concentrate here on raising arc costs. Specifically, we attempt to raise as many arc costs as possible while remaining feasible. Intuitively, increasing the cost of a little-used arc should have a smaller impact on overall cost than increasing the cost of a saturated arc (thus likely allowing other arc costs to be raised). For this reason, we file the arcs for which $\alpha_j^c < \beta_j^c$ in a list ranked in increasing order of utilization (given initially as $f^o(j)/u_j(\beta_j^u)$).

In each step we remove the top arc from the list, increase its cost one level, reoptimize, and verify that the resulting optimal flow still has feasible cost. If not, the cost is returned to the previous level, \bar{k} , and this arc is no longer considered. If the cost is feasible, the increase is retained. At this point, we check if the current arc has zero flow. If so, its cost is raised and its capacity is lowered as much as possible. Otherwise, we lower the capacity of the arc being considered until further reduction would impact that arc's current optimal flow. If after these changes the arc's cost can still be raised, the arc is filed back in the list (again in increasing order of utilization of capacity, now at the current $(\bar{k}|\bar{l})^F$) for a subsequent pass. We then move on to the next arc to be considered. This process continues until the list is empty or until we achieve a cost which is within a specified tolerance of the target cost d .

At that point we have the feasible point $(\bar{k}|\bar{l})^F$, which defines the feasible set

$$F = \{(k|l) \in U : \alpha_j^c \leq k_j \leq \bar{k}_j \text{ for } j \in \mathcal{A} \\ \bar{l}_j \leq l_j \leq \beta_j^u \text{ for } j \in \mathcal{A}\}. \quad (4.1)$$

This feasible state derivation will be used in three different partitioning routines, as we shall describe in Sections 4.2.3 and 4.2.4.

MCDA-I Infeasible State Derivation

Here we seek to derive an infeasible state $(\bar{k}|\bar{l})^I$, starting with the infeasible state $(\beta|\alpha)$, that will cut off as large an infeasible set as is practical using Proposition 2.1.

We begin by finding a minimum cost flow f^o (with infeasible cost) at $(\beta|\alpha)$. All arcs j for which $\alpha_j^c < \beta_j^c$ or $\alpha_j^u < \beta_j^u$ (i.e., all arcs whose cost or capacity can be changed) are filed in a list and are considered in increasing order of utilization, given as $f^o(j)/u_j(\alpha_j^u)$. The reasoning behind this is that in this way we first look at changing the parameters of the least-used arcs. (Recall that we would like for the infeasible point to agree with $(\alpha|\beta)$ in as many positions as possible so as to minimize the number of the resulting undetermined sets.)

At each step we remove the top arc in the list and attempt to lower its cost and raise its capacity one level each. After each of these changes, we reoptimize and check if the resulting state is still infeasible. If not, the previous cost or capacity is restored. But if either of these changes can be retained, and either the cost or the capacity can still be changed (i.e., cost has not yet been lowered to the level α^c or the capacity has not been raised to the level β^u), the arc will be filed back in the list for future consideration, in increasing order of its current utilization. This process continues until no arcs remain to be considered or the MCF cost drops to within a specified tolerance of d .

At that point, we have the infeasible state $(\bar{k}|\bar{l})^I$, which defines the infeasible set

$$I = \{(k|l) \in U : \bar{k}_j \leq k_j \leq \beta_j^c \text{ for } j \in \mathcal{A} \\ \alpha_j^u \leq l_j \leq \bar{l}_j \text{ for } j \in \mathcal{A}\}. \quad (4.2)$$

This infeasible state derivation will be used in three different partitioning routines, as we shall see after presenting alternative feasible and infeasible state derivations in Section 4.2.2.

4.2.2 Decomposition Algorithms II

In this section we discuss an alternative approach to deriving feasible and infeasible cutoff states for use in partitioning algorithms for evaluating P^F , referred to collectively as the Minimum Cost Flow Decomposition Algorithms II (MCDA-II). The feasible and infeasible state derivations given below have the feature that feasible states are derived starting at the infeasible extreme $(\beta|\alpha)$ and infeasible states are derived from the feasible extreme $(\alpha|\beta)$. We describe here the procedures for deriving feasible and infeasible state points in an undetermined interval $U = [\alpha, \beta]$ for which $C(\alpha|\beta) \leq d$ and $C(\beta|\alpha) > d$. Note that, again, α and β are of the form used in Section 4.2.1.

MCDA-II Feasible State Derivation

Here we seek to determine a feasible point $(\bar{k}|\bar{l})^F$ by moving from the infeasible extreme $(\beta|\alpha)$.

We begin by finding f° , a minimum cost flow at $(\beta|\alpha)$. Note that for reasons presented in Section 2.5, we wish to move away from $(\beta|\alpha)$ in as few arcs as possible. Therefore we shall consider arcs in decreasing order of utilization (defined initially as $f^\circ(j)/u_j(\alpha_j^u)$). The reasoning here is that if we first raise capacities and lower costs for heavily-used arcs, we might achieve a feasible cost quickly and leave many arcs set at the $(\beta|\alpha)$ levels. Of course, as soon as a feasible cost is achieved we stop since our goal is then accomplished.

Thus we first identify the most heavily-used arc. If the cost of this arc can be reduced, we reduce it all the way, reoptimize and compute a new MCF cost. If the cost is feasible, we stop. Otherwise, we determine whether or not this arc is still most heavily-used. If it is, we attempt to raise its capacity all the way, again

checking for the effect on the MCF cost. If the current arc is no longer the most attractive arc, we begin anew lowering the cost of the most attractive arc, and the cycle is repeated.

Two practical points bear mentioning here. The first point is that only arcs with positive flow are considered for the changes described above. As a result, the method as stated can stall. That is, we can arrive at a state which still has infeasible cost, and for which all utilized arcs have already been moved as much as possible from $(\beta|\alpha)$. In these situations we lower the costs of unused arcs as much as possible until the situation is remedied (which it must eventually be since we know $(\alpha|\beta)$ is feasible). We first consider such arcs with lowest possible costs and continue in this order until the main method can continue.

The second point is that the determination of most-used arcs and/or lowest-cost unused arcs is done with almost no additional effort. This is so because we determine this information in the course of looping through the arcs to compute the new MCF cost after reoptimizing. Thus, these arcs are always at hand whenever they are needed.

This feasible state determination will be used in three different partitioning routines, as we shall see in Sections 4.2.3 and 4.2.4.

MCDA-II Infeasible State Derivation

Here we seek to derive an infeasible point $(\bar{k}|\bar{l})^I$ by starting at the feasible extreme $(\alpha|\beta)$ and changing costs and/or capacities of only as many arcs as is necessary to achieve a MCF cost exceeding d . We wish for $(\bar{k}|\bar{l})^I$ to agree with $(\alpha|\beta)$ in as many positions as possible so as to minimize the number of the resulting undetermined sets.

We begin by finding f° , a minimum cost feasible flow at $(\alpha|\beta)$. As in MCDA-II feasible decomposition, we identify the most heavily-used arc at the current best point by raising costs and/or lowering capacities in an effort to achieve an infeasible cost. We stop as soon as a cost greater than d is achieved. The same practical

concerns mentioned above (stalling, work done to find new arcs) are also relevant here.

This infeasible state derivation will be used in three different partitioning routines, as we now discuss.

4.2.3 GSD Routines for the Independent Case

In Sections 4.2.1 and 4.2.2 we presented methods for deriving feasible and infeasible cutoff states. These point definitions were made for the purpose of removing feasible and infeasible intervals in accordance with Proposition 2.1. We describe here GSD routines that will use these feasible and infeasible point definitions to evaluate $P^F = P\{C(\mathbf{X}|\mathbf{Y}) \leq d\}$. The efficiency of these routines will be tested Section 4.4. In each case below we describe the partitioning of an undetermined interval $U = [\alpha, \beta]$ for which $C(\alpha|\beta) \leq d$ and $C(\beta|\alpha) > d$.

GSD Feasible Decompositions

The most straightforward feasible decomposition consists of simply deriving a feasible point $(\bar{k}|\bar{l})^F$, removing the resulting feasible set defined by (4.1), and partitioning the remainder of U into at most $2a$ new undetermined intervals. For each arc j we have two undetermined intervals,

$$\begin{aligned} U_j^c = \{(k|l) \in U : & \quad \alpha_i^c \leq k_i \leq \bar{k}_i \quad \text{for } i < j \\ & \quad \bar{k}_j + 1 \leq k_j \leq \beta_j^c \\ & \quad \alpha_i^c \leq k_i \leq \beta_i^c \quad \text{for } i > j \\ & \quad \alpha_i^u \leq l_i \leq \beta_i^u \quad \text{for } i \in \mathcal{A}\} \end{aligned} \quad (4.3)$$

and

$$\begin{aligned} U_j^u = \{(k|l) \in U : & \quad \alpha_i^c \leq k_i \leq \bar{k}_i \quad \text{for } i \in \mathcal{A} \\ & \quad \bar{l}_i \leq l_i \leq \beta_i^u \quad \text{for } i < j \\ & \quad \alpha_j^u \leq l_j \leq \bar{l}_j - 1 \end{aligned} \quad (4.4)$$

$$\alpha_i^u \leq l_i \leq \beta_i^u \quad \text{for } i > j\}.$$

But in many cases we wish to also remove infeasible sets in each iteration. Section 2.5 suggests that this is best accomplished by finding limiting feasible indices \hat{k}_j and \hat{l}_j as given in Definition 2.8. Note that some of these are already known. If $\bar{k}_j = \beta_j^c$ for any j then $\hat{k}_j = \beta_j^c$. Similarly, $\hat{l}_j = \alpha_j^u$ if $\bar{l}_j = \alpha_j^u$. After we find all $2a$ indices, the standard GSD feasible decomposition of Section 2.4 removes up to $2a$ infeasible sets of the form

$$\begin{aligned} I_j^c = \{(k|l) \in U : & \quad \alpha_i^c \leq k_i \leq \hat{k}_i & \text{for } i < j \\ & \hat{k}_j + 1 \leq k_j \leq \beta_j^c & \\ & \alpha_i^c \leq k_i \leq \beta_i^c & \text{for } i > j \\ & \alpha_i^u \leq l_i \leq \beta_i^u & \text{for } i \in \mathcal{A}\} \end{aligned} \quad (4.5)$$

and

$$\begin{aligned} I_j^u = \{(k|l) \in U : & \quad \alpha_i^c \leq k_i \leq \hat{k}_i & \text{for } i \in \mathcal{A} \\ & \hat{l}_i \leq l_i \leq \beta_i^u & \text{for } i < j \\ & \alpha_j^u \leq l_j \leq \hat{l}_j - 1 & \\ & \alpha_i^u \leq l_i \leq \beta_i^u & \text{for } i > j\} \end{aligned} \quad (4.6)$$

and up to $2a$ new undetermined intervals given by

$$\begin{aligned} U_j^c = \{(k|l) \in U : & \quad \alpha_i^c \leq k_i \leq \bar{k}_i & \text{for } i < j \\ & \bar{k}_j + 1 \leq k_j \leq \hat{k}_j & \\ & \alpha_i^c \leq k_i \leq \hat{k}_i & \text{for } i > j \\ & \hat{l}_i \leq l_i \leq \beta_i^u & \text{for } i \in \mathcal{A}\} \end{aligned} \quad (4.7)$$

and

$$\begin{aligned} U_j^u = \{(k|l) \in U : & \quad \alpha_i^c \leq k_i \leq k_i & \text{for } i \in \mathcal{A} \\ & \bar{l}_i \leq l_i \leq \beta_i^u & \text{for } i < j \end{aligned}$$

$$\hat{l}_j \leq l_j \leq \bar{l}_j - 1 \quad (4.8)$$

$$\hat{l}_i \leq l_i \leq \beta_i^u \quad \text{for } i > j\}.$$

Thus, using MCDA-I or MCDA-II feasible point derivations with or without limiting feasible indices gives us four GSD feasible routines, each of which follows the basic form of what was called Pure Option 1 in Section 2.4.

GSD Infeasible Decompositions

As with feasible decomposition, we have the option of using limiting indices or not. If we decline to do so, then we simply derive the infeasible state $(\bar{k}|\bar{l})^I$ by one of the two methods given above, remove the infeasible set I given in (4.2), and partition the remainder of U into up to $2a$ undetermined intervals given by

$$\begin{aligned} U_j^c = \{(k|l) \in U : & \quad \bar{k}_i \leq k_i \leq \beta_i^c \quad \text{for } i < j \\ & \alpha_j^c \leq k_j \leq \bar{k}_j - 1 \\ & \alpha_i^c \leq k_i \leq \beta_i^c \quad \text{for } i > j \\ & \alpha_i^u \leq l_i \leq \beta_i^u \quad \text{for } i \in \mathcal{A}\} \end{aligned} \quad (4.9)$$

and

$$\begin{aligned} U_j^u = \{(k|l) \in U : & \quad \bar{k}_i \leq k_i \leq \beta_i^c \quad \text{for } i \in \mathcal{A} \\ & \alpha_i^u \leq l_i \leq \bar{l}_i \quad \text{for } i < j \\ & \bar{l}_j + 1 \leq l_j \leq \beta_j^u \\ & \alpha_i^u \leq l_i \leq \beta_i^u \quad \text{for } i > j\}. \end{aligned} \quad (4.10)$$

If we choose to also remove feasible sets during each iteration, we first determine limiting infeasible indices \hat{k}_j and \hat{l}_j as in Definition 2.9. (Again, if $\bar{k}_j = \alpha_j^c$ we may set $\hat{k}_j = \alpha_j^c$ and if $\bar{l}_j = \beta_j^u$ we may set $\hat{l}_j = \beta_j^u$.) Once all indices are determined, we may remove up to $2a$ feasible sets of the form

$$F_j^c = \{(k|l) \in U : \quad \hat{k}_i \leq k_i \leq \beta_i^c \quad \text{for } i < j$$

$$\alpha_j^c \leq k_j \leq \hat{k}_j - 1 \quad (4.11)$$

$$\alpha_i^c \leq k_i \leq \beta_i^c \quad \text{for } i > j$$

$$\alpha_i^u \leq l_i \leq \beta_i^u \quad \text{for } i \in \mathcal{A}$$

and

$$\begin{aligned} F_j^u = \{(k|l) \in U : & \quad \hat{k}_i \leq k_i \leq \beta_i^c \quad \text{for } i \in \mathcal{A} \\ & \quad \alpha_i^u \leq l_i \leq \hat{l}_i \quad \text{for } i < j \\ & \quad \hat{l}_j + 1 \leq l_j \leq \beta_j^u \\ & \quad \alpha_i^u \leq l_i \leq \beta_i^u \quad \text{for } i > j\} \end{aligned} \quad (4.12)$$

and partition the remainder of U into up to $2a$ new undetermined sets given by

$$\begin{aligned} U_j^c = \{(k|l) \in U : & \quad \bar{k}_i \leq k_i \leq \beta_i^c \quad \text{for } i < j \\ & \quad \hat{k}_j \leq k_j \leq \bar{k}_j - 1 \\ & \quad \hat{k}_i \leq k_i \leq \beta_i^c \quad \text{for } i > j \\ & \quad \alpha_i^u \leq l_i \leq \hat{l}_i \quad \text{for } i \in \mathcal{A}\} \end{aligned} \quad (4.13)$$

and

$$\begin{aligned} U_j^u = \{(k|l) \in U : & \quad \bar{k}_i \leq k_i \leq \beta_i^c \quad \text{for } i \in \mathcal{A} \\ & \quad \alpha_i^u \leq l_i \leq \bar{l}_i \quad \text{for } i < j \\ & \quad \bar{l}_j + 1 \leq l_j \leq \hat{l}_j \\ & \quad \alpha_i^u \leq l_i \leq \hat{l}_i^u \quad \text{for } i > j\}. \end{aligned} \quad (4.14)$$

Thus we have four different GSD infeasible decomposition routines, each following the general description of Pure Option 2 in Section 2.4.

4.2.4 Hybrid Decomposition Routine for the Independent Case

In Section 2.5 we were cautioned that deviating from GSD would most likely result in diminished performance since such a departure was proved to cause an

unnecessary proliferation of undetermined intervals. Nevertheless, since many other factors play a part in determining a routine's ultimate effectiveness, it is worthwhile to attempt other kinds of partitioning schemes. In that spirit, we describe here a routine that is intuitively very appealing, but that will file up to $4a - 1$ new undetermined intervals in each iteration (instead of up to $2a$ as in GSD routines). This procedure, which we have called a *hybrid* routine, removes the largest practical feasible interval and the largest practical infeasible interval in each iteration. Specifically, in this routine we determine a feasible state $(\bar{k}|\bar{l})^F$ by either MCDA-I or MCDA-II feasible state definition and an infeasible state $(\bar{k}|\bar{l})^I$ by the corresponding infeasible state definition. We remove the resulting feasible set F as given in (4.1), adding its probability to the accumulated lower bound for P^F , and the infeasible set I as given in (4.2), and decompose the remainder of U in the manner described in the proof of Proposition 2.3.

Thus we have two different hybrid routines (one corresponding to MCDA-I and one corresponding to MCDA-II).

We now conclude our discussion of the case of independent arcs by describing the algorithm MCDIST.

4.2.5 Computing the Entire Distribution of MCF Costs

The algorithms presented thus far are well-suited to situations in which a limit on the cost of an optimal flow exists *a priori* (e.g., budgetary constraints, availability of materials, imposed specifications). If that is not the case, it is useful to completely characterize the probabilistic behavior of $C(\mathbf{X}|\mathbf{Y})$, including the computation of the mean and higher moments of $C(\mathbf{X}|\mathbf{Y})$. In this section we describe MCDIST, a routine that provides precisely this information by producing the probability distribution of $C(\mathbf{X}|\mathbf{Y})$.

In a slight departure from straight GSD, MCDIST is not concerned with calculating a single probability measure. Rather, it is concerned with computing the probability function $q(d) = P\{C(\mathbf{X}|\mathbf{Y}) = d\}$, $d \geq 0$. Lower bounds for these values,

denoted $q_i(d)$, are initialized to zero and accumulate the appropriate probabilities as decomposition proceeds. In the following description, \mathcal{U} consists of those undetermined sets not yet considered. As with the procedure STDIST, we assume that MCF costs will be integral (which we enforce by requiring integral cost and capacity values).

Procedure MCDIST

Step 1 Find $C_{lo} = C(1, \dots, 1 | n_1, \dots, n_a)$ and $C_{hi} = C(m_1, \dots, m_a | 1, \dots, 1)$. For $d = C_{lo}, \dots, C_{hi}$, set $q_i(d) = 0$. Set $\mathcal{U} = \emptyset$ and $U = \Omega$.

Step 2 Let α and β denote, respectively, the lower and upper endpoints of U . Find f^o , a minimum cost flow at $(\alpha|\beta)$, with cost $C(\alpha|\beta)$.

Step 3 Let $(\bar{k}|\bar{l})$ be derived from $(\alpha|\beta)$ by pushing costs and capacities of unused arcs to the opposite extreme and reducing unused capacities.

Step 4 Set $q_i(C(\alpha|\beta)) = q_i(C(\alpha|\beta)) + P\{F\}$, where F is as defined by (4.1).

Step 5 For $j \in \mathcal{A}$: If $\bar{k}_j \neq \beta_j^c$, file the interval U_j^c defined by (4.3) in \mathcal{U} . If $\bar{l}_j \neq \alpha_j^u$, file the interval U_j^u defined by (4.4) in \mathcal{U} .

Step 6 If $\mathcal{U} = \emptyset$, stop. Otherwise, remove a set U from \mathcal{U} and go to step 2.

Note that in step 3 there really isn't room for improving $(\bar{k}|\bar{l})$ (i.e., for pushing it further from $(\alpha|\beta)$) since increasing the cost of an arc which carries flow necessarily changes the cost of a minimum cost flow.

As usual, MCDIST may be terminated prior to complete partitioning of Ω to produce bounds. One can stop, for example, when the bounds are sufficiently close or when a predetermined number of sets have been decomposed. To produce bounds on $F(d) = P\{C(\mathbf{X}|\mathbf{Y}) \leq d\}$ for all d , we substitute step $\tilde{6}$ for step 6, and add steps 7 and 8.

Step 6 If $\mathcal{U} = \emptyset$, construct the cumulative distribution function, and stop. If a stopping condition is met, go to step 7. Otherwise, remove a set U from \mathcal{U} and go to step 2.

Step 7 For $d = C_{l_0}, \dots, C_{h_i}$: let $F_l(d) = F_u(d) = \sum_{i=C_{l_0}}^d q(i)$.

Step 8 For all $U \in \mathcal{U}$: Let α and β denote, respectively, the lower and upper endpoints of U . Find $C(\alpha|\beta)$ and $C(\beta|\alpha)$.

For $C(\alpha|\beta) \leq d < C(\beta|\alpha)$: Set $F_u(d) = F_u(d) + P\{U\}$.

For $C(\beta|\alpha) \leq d \leq C_{h_i}$: Set $F_u(d) = F_u(d) + P\{U\}$ and $F_l(d) = F_l(d) + P\{U\}$.

The performance of this algorithm will be evaluated in Section 4.4.

4.3 The Case of Dependent Costs and Capacities

Thus far we have considered minimum cost flow problems in which arc costs and capacities are given as mutually independent random variables. In practice, though, it is sometimes the case that the cost and upper bound for a given arc vary together.

Example 4.1 A link in a transportation network can move 20 units of commodity per unit time at an average cost of \$.15 per unit when it is in perfect condition. The link can sustain minor damage and still operate with diminished capacity of 15 units and a moderate increase in cost to \$.20 (where the increased cost is due to additional required monitoring of the link). Serious damage to the link causes its capacity to drop to only 5 units and increases the average cost per unit to \$.30 since at that point a worker is dispatched to make repairs on the link. Finally, the link can be completely failed (its capacity is reduced to 0 units).

In order to model situations such as that given in Example 4.1, we associate with each arc a vector-valued random variable $Z_j = (X_j, Y_j)$, where X_j and Y_j represent cost and upper bound, respectively, as in Section 4.2, but where X_j is a deterministic function of Y_j . In Example 4.1, Z_j takes on values $(.15, 20)$, $(.20, 15)$, $(.30, 5)$ and

$(\cdot, 0)$. (The cost per unit associated with capacity 0 is immaterial.) As mentioned earlier, we require that $c_j(1) \leq \dots \leq c_j(n_j)$ and $u_j(1) \geq \dots \geq u_j(n_j)$ so that we may classify Z_j as a lower feasible variable. The state space Ω of $\mathbf{Z} = (Z_1, \dots, Z_a)$ consists of $\prod_{j=1}^a n_j$ state points of the form $z = ((c_1(l_1), u_1(l_1)), \dots, (c_a(l_a), u_a(l_a)))$, where $l_j \in \{1, \dots, n_j\}$ for each $j \in \mathcal{A}$. As before, we shall refer to this state point as $l = (l_1, \dots, l_a)$.

In the remainder of this section, we present a number of algorithms for solving problem MC2. To the extent possible, they will largely be adaptations of the routines given in Section 4.2 for the case of independent costs and capacities. However, we shall see that some flexibility is lost in translating those ideas into the present setting. We will now be unable to change *only* the cost or *only* the capacity of an arc. For example, we cannot reduce unused arc capacity without impacting an optimal solution since that arc's cost will necessarily increase.

In Section 4.3.1 we give dependent versions of feasible and infeasible state derivations corresponding to MCDA-I. Section 4.3.2 gives derivations for the dependent MCDA-II. GSD routines for solving the present problem are given in Section 4.3.3, and "hybrid" algorithms are given in Section 4.3.4. In Section 4.3.5 we present the dependent version of MCDIST for computing the entire distribution of $C(\mathbf{Z})$. The performance of all these routines will be evaluated in Section 4.4.

4.3.1 Decomposition Algorithms I

We present here feasible and infeasible state derivations to be used in a number of partitioning algorithms for solving problem MC2. We seek to partition an undetermined interval $U = [\alpha, \beta]$ for which $C(\alpha) \leq d$ and $C(\beta) > d$. These derivations have the common feature that feasible cutoff states will be found by starting at the feasible extreme α (which corresponds to lowest cost paired with highest capacity for each arc) and infeasible cutoff states will be derived from the infeasible extreme β (highest cost paired with lowest capacity).

MCDA-I Feasible State Derivation

Here we wish to derive a feasible state \bar{l}^F by moving from the extreme feasible point α . We begin by finding f° , a minimum cost feasible flow at α . As with the independent MCDA-I, we may simultaneously raise the arc cost and lower the arc capacity of any arc that does not carry flow in f° without impacting optimality. (Recall that cost and capacity form a dependent pair so that this operation consists of moving to higher-indexed state vectors.) After doing this, we file the remaining arcs j (for which $\alpha_j < \beta_j$) in a heap sorted in increasing order of utilization (given as $f^\circ(j)/u_j(\alpha_j)$). As explained before, we do this so as to make \bar{l}_j^F close to β_j for as many arcs as possible to cut down on the number of undetermined sets filed.

We consider arcs in this order and attempt to simultaneously increase arc cost and reduce arc capacity a single level. After each such move, we re-optimize and check the new MCF cost. If the move has caused the cost to exceed d , we restore the previous cost and capacity level and eliminate the arc from further consideration. Otherwise, the change is retained and we re-file the arc (as long as we have not raised its cost and lowered its capacity as much as possible) for a future pass. This process continues until no arcs remain to be considered or the MCF cost is raised to within a specified tolerance of the target cost d .

MCDA-I Infeasible State Derivation

Here we seek to derive an infeasible point \bar{l}^I by moving from the infeasible extreme β . Our goal is to use this point to cut off as large a feasible set as possible without causing unnecessary proliferation of new undetermined sets. As such, we will again try to move as many arcs as possible toward the level α by first considering little-used arcs and proceeding in order of increased utilization.

We begin by finding f° , a minimum cost flow (with cost exceeding d) at the point β . We then consider arcs in increasing order of utilization. For each arc considered, we lower its cost and raise its capacity a single level, keeping this change only if the

resulting MCF still has infeasible cost. If that is the case, and there is still room to move this arc's cost and capacity, we file it back in a list for future passes. This process continues until there are no more arcs to consider or the MCF cost falls to within a specified tolerance of d .

4.3.2 Decomposition Algorithms II

This section contains feasible and infeasible state derivations to be used in partitioning algorithms for solving problem MC2. The descriptions below give details for finding feasible and infeasible cutoff states for an undetermined set $U = [\alpha, \beta]$ having $C(\alpha) \leq d$ and $C(\beta) > d$. Feasible states will be derived starting at the infeasible extreme β and infeasible states will be derived from α . The motivation for this approach is to produce feasible cutoff states that agree with β in as many positions as possible and infeasible cutoff states close to α . As such, since we wish to move few arcs away from their starting extreme, we shall consider arcs in decreasing order of utilization.

MCDA-II Feasible State Derivation

Here we derive a feasible point \bar{l}^F from the infeasible extreme β . We begin by finding f° , a minimum cost flow (with cost exceeding d) at the point β . We wish to look first at heavily-used arcs. As in the independent version of MCDA-II, at all times we have available the current most utilized arc. We reduce the cost and increase the capacity of this arc all the way, and then re-optimize to determine the effect of this change on the MCF cost. We continue in this fashion until reoptimization reveals that a feasible cost is achieved. At that point we stop.

MCDA-II Infeasible State Derivation

We seek to derive an infeasible state point \bar{l}^I by starting at the feasible extreme α . The method by which we do this is essentially the same as that given above for feasible states. Specifically, we find a minimum cost feasible flow f° at α and then

move away from this point in as few arcs as possible. We accomplish this by always considering the most heavily-used arc. We stop as soon as a MCF cost exceeding d has been achieved.

4.3.3 GSD Routines for the Dependent Case

In Sections 4.3.1 and 4.3.2 we described techniques for deriving feasible and infeasible states for the minimum cost flow problem with dependent arc costs and capacities. These point definitions were made for the purpose of removing feasible and infeasible intervals in partitioning algorithms for solving problem MC2. We describe here GSD routines for this problem. The examples in Section 4.4 will test the performance of these routines. In each case below we detail the partitioning of an undetermined interval $U = [\alpha, \beta]$ for which $C(\alpha) \leq d$ and $C(\beta) > d$.

GSD Feasible Decompositions

We describe first the simplest GSD feasible decomposition, in which either MCDA-I or MCDA-II is used to derive a feasible state \bar{l}^F , the feasible interval

$$F = \{l \in U : \alpha_j \leq l_j \leq \bar{l}_j \text{ for } j \in \mathcal{A}\} \quad (4.15)$$

is removed, and the remainder of U is split into up to a undetermined intervals given by

$$\begin{aligned} U_j = \{l \in U : & \alpha_i \leq l_i \leq \bar{l}_i \quad \text{for } i < j \\ & \bar{l}_j + 1 \leq l_j \leq \beta_j \\ & \alpha_i \leq l_i \leq \beta_i \quad \text{for } i > j\}. \end{aligned} \quad (4.16)$$

However, since the dependence of cost and capacity limit the depth to which we may push a feasible state, we generally choose to also remove infeasible sets by defining limiting feasible indices \hat{l}_j as given in Definition 2.8. With the a indices identified, we remove F as above and up to a infeasible intervals

$$I_j = \{l \in U : \alpha_i \leq l_i \leq \hat{l}_i \text{ for } i < j$$

$$\hat{l}_j + 1 \leq l_j \leq \beta_j \quad (4.17)$$

$$\alpha_i \leq l_i \leq \beta_i \quad \text{for } i > j\},$$

partitioning the remainder into up to a undetermined sets given by

$$\begin{aligned} U_j = \{l \in U : & \quad \alpha_i \leq l_i \leq \bar{l}_i \quad \text{for } i < j \\ & \quad \bar{l}_j + 1 \leq l_j \leq \hat{l}_j \\ & \quad \alpha_i \leq l_i \leq \hat{l}_i \quad \text{for } i > j\}. \end{aligned} \quad (4.18)$$

Thus, using either MCDA-I or MCDA-II with or without limiting feasible indices yields four GSD routines, each following the basic form of Pure Option 1 described in Section 2.4.

GSD Infeasible Decompositions

As with feasible decomposition, we have the option of using limiting infeasible indices or not. If we decline to do so, then we simply derive the infeasible state \bar{l} by one of the two methods presented, remove the infeasible set

$$I = \{l \in U : \bar{l}_j \leq l_j \leq \beta_j \quad \text{for } j \in \mathcal{A}\} \quad (4.19)$$

and divide the remainder of U into up to a undetermined sets

$$\begin{aligned} U_j = \{l \in U : & \quad \bar{l}_i \leq l_i \leq \beta_i \quad \text{for } i < j \\ & \quad \alpha_j \leq l_j \leq \bar{l}_j - 1 \\ & \quad \alpha_i \leq l_i \leq \beta_i \quad \text{for } i > j\}. \end{aligned} \quad (4.20)$$

If we choose to also remove feasible sets during each iteration, we first find limiting infeasible indices \hat{l}_j given in Definition 2.9. In that case, I is removed as above along with up to a feasible intervals

$$\begin{aligned} F_j = \{l \in U : & \quad \hat{l}_i \leq l_i \leq \beta_i \quad \text{for } i < j \\ & \quad \alpha_j \leq l_j \leq \hat{l}_j - 1 \\ & \quad \alpha_i \leq l_i \leq \beta_i \quad \text{for } i > j\}, \end{aligned} \quad (4.21)$$

and up to a undetermined sets given by

$$\begin{aligned}
 U_j = \{l \in U : & \quad \bar{l}_i \leq l_i \leq \beta_i \quad \text{for } i < j \\
 & \quad \hat{l}_j \leq l_j \leq \bar{l}_j - 1 \\
 & \quad \alpha_i \leq l_i \leq \hat{l}_i \quad \text{for } i > j\}.
 \end{aligned}
 \tag{4.22}$$

Thus we have four GSD infeasible decomposition schemes, each following the general description of Pure Option 2 in Section 2.4.

4.3.4 Hybrid Decomposition Routine for the Dependent Case

As in the case of independent costs and capacities, in each iteration we attempt to remove from the current undetermined interval $U = [\alpha, \beta]$ a single arbitrary feasible set and a single arbitrary infeasible set. The tradeoff to this intuitive idea (removing as much of the feasible and infeasible regions as possible) is that the remainder of U can require as many as $2a - 1$ undetermined intervals for its partition.

Basically, this *hybrid* routine derives a feasible point \bar{l}^F by either MCDA-I or MCDA-II feasible state derivation and an infeasible state \bar{l}^I by the corresponding infeasible derivation. The probability of the feasible set F (as given in (4.15)) is added to the lower bound for P^F , the infeasible set I (as in (4.15)) is discarded, and $U \setminus (F \cup I)$ is partitioned as in the proof of Proposition 2.3.

4.3.5 Computing the Entire Distribution of MCF Costs

As we indicated earlier, it is often useful to derive the entire distribution of $C(\mathbf{Z})$. The procedure here is completely analogous to that given in Section 4.2.5 with the exception of the derivation of our feasible point in step 3. The only move we can make in pushing in from α while keeping the MCF cost constant at $C(\alpha)$ is with the unused arcs. We cannot decrease excess capacity of arcs that carry flow since this would also increase the arc cost and hence the MCF cost.

With \bar{I}^F thus defined, we remove the feasible set F as in (4.15) and up to a undetermined sets U_j defined by (4.16).

4.4 Stochastic MCF Examples

In this section we demonstrate the performance of the various routines developed in the present chapter. We test eight GSD routines, two hybrid routines and MCDIST both for the case of independent arc costs and capacities and for the case of dependent pairs of arc costs and capacities. In each case, we label GSD routines that do not use limiting indices by "1" and those using limiting indices by "2." Our main goal is to compare the various approaches and evaluate their efficiency in producing bounds. All codes were implemented in FORTRAN 77, and were compiled and executed on a SUN SPARCstation 2. CPU times reported are in seconds. Minimum cost flows were found using the relaxation codes RELAXT and SENSTV, developed by Bertsekas and Tseng [9]. In all applications, the list of undetermined sets was kept as a heap sorted by probability until the most probable undetermined set had probability less than 10^{-20} . In the tables below we denote the sum of the probabilities of the undetermined intervals remaining in the list \mathcal{U} (the difference between upper and lower bounds) by $P\{\mathcal{U}\}$.

Example 4.2 (Example 1.3 revisited) The network in Table 4.1 has 10 nodes and 21 directed arcs. Each arc has between two and four possible costs and between one and four possible capacities. The cardinality of the state space is roughly 9×10^{18} . We wish to ship 15 units from node $s = 1$ to node $t = 10$. The range of possible optimal flow costs is from 1591 to 3718. Note that this data set differs from most we have used in that it does not have probability as heavily concentrated at the *status quo*. Later in this section, we re-evaluate this problem with alternative input distributions. We use the ten decomposition routines described above to compute $P\{C(X) \leq 1900\}$ and $P\{C(X) \leq 3500\}$. Results for $F(1900)$ are in Table 4.3, for $F(3500)$ in Table 4.4. We see that hybrid routines did not perform as well as the

Table 4.1: Arc Cost Distributions for Example 4.2.

Arc	Cost (Probability)			
1 = (1,2)	70 (0.5)	73 (0.3)	94 (0.2)	
2 = (1,3)	25 (0.5)	35 (0.4)	82 (0.1)	
3 = (1,4)	42 (0.5)	48 (0.3)	61 (0.2)	
4 = (2,5)	26 (0.4)	31 (0.3)	55 (0.2)	88 (0.1)
5 = (2,6)	58 (0.4)	70 (0.3)	95 (0.3)	
6 = (3,2)	15 (0.6)	73 (0.4)		
7 = (3,7)	65 (0.5)	74 (0.4)	75 (0.1)	
8 = (3,8)	59 (0.6)	72 (0.3)	98 (0.1)	
9 = (4,3)	21 (0.3)	32 (0.3)	85 (0.2)	98 (0.2)
10 = (4,9)	89 (0.7)	96 (0.3)		
11 = (5,7)	32 (0.6)	48 (0.2)	67 (0.2)	
12 = (5,10)	63 (0.5)	99 (0.5)		
13 = (6,3)	66 (0.8)	85 (0.1)	98 (0.1)	
14 = (6,5)	6 (0.4)	15 (0.3)	39 (0.2)	58 (0.1)
15 = (6,7)	2 (0.6)	48 (0.4)		
16 = (7,8)	16 (0.3)	18 (0.3)	40 (0.2)	52 (0.2)
17 = (7,10)	16 (0.5)	34 (0.4)	71 (0.1)	
18 = (8,4)	90 (0.5)	96 (0.5)		
19 = (8,9)	17 (0.4)	49 (0.4)	53 (0.1)	65 (0.1)
20 = (9,7)	3 (0.5)	30 (0.4)	50 (0.1)	
21 = (9,10)	6 (0.5)	12 (0.3)	54 (0.1)	66 (0.1)

better corresponding GSD routines. MCDIST suffered badly on this example. After decomposing 10,000,000 sets, the probability of the remaining undetermined sets exceeded 0.80. This is due to the tremendous size of the state space (as mentioned above) and to the nature of the input distributions.

Example 4.3 Taking the input distribution for Example 4.2, we make two changes to make it more reasonable. First, we bring the possible values for each arc cost and arc capacity random variable closer together. Next, we adjust the probabilities so that in each case the *status quo* is more heavily weighted than before. The cardinality of the state space remains the same as that of Example 4.2. The resulting

Table 4.2: Arc Capacity Distributions for Example 4.2.

Arc	Capacity (Probability)			
1 = (1,2)	10 (0.3)	15 (0.7)		
2 = (1,3)	7 (0.2)	10 (0.2)	15 (0.6)	
3 = (1,4)	8 (0.5)	13 (0.5)		
4 = (2,5)	6 (0.2)	10 (0.3)	17 (0.5)	
5 = (2,6)	4 (0.1)	7 (0.2)	8 (0.3)	10 (0.4)
6 = (3,2)	7 (0.1)	9 (0.2)	15 (0.7)	
7 = (3,7)	5 (0.1)	8 (0.1)	10 (0.3)	16 (0.5)
8 = (3,8)	8 (0.1)	12 (0.9)		
9 = (4,3)	9 (0.1)	13 (0.2)	18 (0.7)	
10 = (4,9)	4 (0.2)	7 (0.2)	11 (0.2)	113 (0.4)
11 = (5,7)	8 (0.5)	14 (0.5)		
12 = (5,10)	6 (0.1)	15 (0.2)	17 (0.7)	
13 = (6,3)	4 (0.1)	7 (0.2)	12 (0.2)	20 (0.5)
14 = (6,5)	5 (0.3)	12 (0.7)		
15 = (6,7)	5 (0.2)	7 (0.3)	8 (0.5)	
16 = (7,8)	5 (1.0)			
17 = (7,10)	6 (0.2)	14 (0.8)		
18 = (8,4)	3 (0.1)	5 (0.1)	9 (0.2)	13 (0.6)
19 = (8,9)	8 (0.2)	10 (0.2)	14 (0.6)	
20 = (9,7)	4 (0.1)	7 (0.2)	9 (0.3)	12 (0.4)
21 = (9,10)	5 (0.3)	12 (0.7)		

Table 4.3: Computing $F(1900) \approx 0.3037859483$ for Example 4.2.

Routine	Full Decomposition		10,000 Sets Decomposed				
	Sets	Time	Bounds		Time	$ \mathcal{U} $	$P\{\mathcal{U}\}$
MCDA-I							
-Feasible 1	78,488	310.25	0.29394898	0.31171238	40.26	4181	.0178
-Feasible 2	225,270	699.59	0.28139953	0.33257054	25.40	2500	.0512
-Infeasible 1	> 500,000		0.00000000	0.64673483	45.44	1925	.6467
-Infeasible 2	> 500,000		0.00000000	0.64673483	22.63	1924	.6467
-Hybrid	> 500,000		0.21396513	0.47207714	33.32	2525	.2581
MCDA-II							
-Feasible 1	> 500,000		0.09918489	0.51638467	28.19	4463	.4172
-Feasible 2	> 500,000		0.04420176	0.87545531	19.28	4513	.8313
-Infeasible 1	> 500,000		0.13529266	0.39088202	26.82	3744	.2556
-Infeasible 2	> 500,000		0.04796162	0.40547247	15.94	3727	.3566
-Hybrid	> 500,000		0.12020747	0.54014936	16.91	3786	.4200

Table 4.4: Computing $F(3500) = 0.9999996197$ for Example 4.2.

Routine	Full Decomposition		10,000 Sets Decomposed				
	Sets	Time	Bounds		Time	$ \mathcal{U} $	$P\{\mathcal{U}\}$
MCDA-I							
-Feasible 1	132,944	443.50	0.99999805	0.99999989	92.07	3411	.000002
-Feasible 2	201,895	507.75	0.99999798	0.99999990	77.16	3413	.000002
-Infeasible 1	> 500,000		0.99998046	1.00000000	86.03	4369	.000020
-Infeasible 2	> 500,000		0.99768438	1.00000000	97.59	3696	.002316
-Hybrid	> 500,000		0.99940423	1.00000000	71.06	2762	.000596
MCDA-II							
-Feasible 1	> 500,000		0.99998638	1.00000000	59.16	2533	.000014
-Feasible 2	> 500,000		0.99998638	1.00000000	44.53	2530	.000014
-Infeasible 1	> 500,000		0.99999060	0.99999993	57.81	4725	.000009
-Infeasible 2	> 500,000		0.99763126	0.99999999	39.88	3265	.002369
-Hybrid	> 500,000		0.99965996	0.99999999	39.34	3381	.000340

distribution is listed in Tables 4.5 and 4.6. The possible optimal flow costs now range from 1591 to 2664. We evaluate $F(1800)$ in Table 4.7. Similar results for the evaluation of $F(2000)$ are found in Table 4.8. All ten routines required more than 500,000 iterations for the exact evaluation of $F(2000)$, too many to be of practical use. However, the bounds very quickly achieved a useable width. Despite the change in the data file, MCDIST still labored. After 1 million iterations, the probability of the remaining undetermined sets was 0.35. After 10 million, 0.33 still remained. A partial listing of the result of 10 million iterations is given in Table 4.9. Notice that bounds for small values of d are relatively tight as compared with those for larger d . This is due to the fact that procedure MCDIST begins at the lowest possible MCF cost and moves up. Bounds on the mean optimal flow cost, calculated from the bounds on this distribution, are 1735.4 to 1842.3.

Example 4.4 Here the cost and capacity of each arc form a dependent pair, as shown in Table 4.10. The cardinality of the state space is about 8 billion. The range of optimal flow costs is from 1591 to 3607. In Table 4.11 are results for $F(1900)$. The hybrid routines clearly required more computational effort than the best GSD routines. Table 4.12 contains results for $F(3000)$ while Table 4.13 contains a partial listing of the cdf $F(d)$. Full decomposition considered 1,128,522 sets. The mean optimal flow cost computed from this distribution is 1979.05, with standard deviation 251.9.

Example 4.5 Just as we turned Example 4.2 into Example 4.3, here we have taken the data from Example 4.4 and created the present example. Again, the spread of arc costs and capacities was reduced and the probabilities were shifted so that significantly more weight is on the *status quo*. The resulting distribution, given in Table 4.14, gives a range of possible optimal flow costs from 1591 to 2586. In Table 4.15 we summarize results for evaluating $F(1650)$. Results for $F(2000)$ are in Table 4.16. Again, the hybrid routines took significantly longer to converge than the other

Table 4.5: Arc Cost Distributions for Example 4.3.

Arc	Cost (Probability)			
1 = (1,2)	70 (0.8)	73 (0.1)	84 (0.1)	
2 = (1,3)	25 (0.9)	35 (0.05)	42 (0.05)	
3 = (1,4)	42 (0.8)	48 (0.1)	51 (0.1)	
4 = (2,5)	26 (0.7)	31 (0.1)	45 (0.1)	50 (0.1)
5 = (2,6)	58 (0.8)	70 (0.1)	85 (0.1)	
6 = (3,2)	15 (0.9)	23 (0.1)		
7 = (3,7)	65 (0.7)	74 (0.2)	75 (0.1)	
8 = (3,8)	59 (0.8)	65 (0.1)	78 (0.1)	
9 = (4,3)	21 (0.7)	32 (0.1)	45 (0.1)	50 (0.1)
10 = (4,9)	89 (0.8)	96 (0.2)		
11 = (5,7)	32 (0.8)	48 (0.1)	57 (0.1)	
12 = (5,10)	63 (0.8)	79 (0.2)		
13 = (6,3)	66 (0.8)	75 (0.1)	88 (0.1)	
14 = (6,5)	6 (0.6)	15 (0.2)	19 (0.1)	28 (0.1)
15 = (6,7)	2 (0.8)	8 (0.2)		
16 = (7,8)	16 (0.7)	18 (0.1)	22 (0.1)	35 (0.1)
17 = (7,10)	16 (0.8)	24 (0.1)	31 (0.1)	
18 = (8,4)	90 (0.9)	96 (0.1)		
19 = (8,9)	17 (0.7)	29 (0.1)	33 (0.1)	45 (0.1)
20 = (9,7)	3 (0.7)	10 (0.2)	15 (0.1)	
21 = (9,10)	6 (0.6)	12 (0.2)	24 (0.1)	26 (0.1)

Table 4.6: Arc Capacity Distributions for Example 4.3.

Arc	Capacity (Probability)			
1 = (1,2)	10 (0.1)	12 (0.9)		
2 = (1,3)	7 (0.1)	10 (0.2)	15 (0.7)	
3 = (1,4)	8 (0.2)	13 (0.8)		
4 = (2,5)	6 (0.1)	10 (0.1)	12 (0.8)	
5 = (2,6)	4 (0.1)	7 (0.1)	8 (0.2)	10 (0.6)
6 = (3,2)	7 (0.1)	9 (0.2)	15 (0.7)	
7 = (3,7)	5 (0.1)	8 (0.1)	10 (0.1)	16 (0.7)
8 = (3,8)	8 (0.1)	12 (0.9)		
9 = (4,3)	9 (0.1)	13 (0.2)	18 (0.7)	
10 = (4,9)	4 (0.1)	7 (0.1)	11 (0.2)	13 (0.6)
11 = (5,7)	8 (0.1)	14 (0.9)		
12 = (5,10)	6 (0.1)	15 (0.2)	17 (0.7)	
13 = (6,3)	4 (0.1)	7 (0.1)	12 (0.1)	18 (0.7)
14 = (6,5)	5 (0.1)	12 (0.9)		
15 = (6,7)	5 (0.1)	7 (0.1)	8 (0.8)	
16 = (7,8)	5 (1.0)			
17 = (7,10)	6 (0.2)	14 (0.8)		
18 = (8,4)	3 (0.1)	5 (0.1)	9 (0.2)	13 (0.6)
19 = (8,9)	8 (0.1)	10 (0.2)	14 (0.7)	
20 = (9,7)	4 (0.1)	7 (0.1)	9 (0.2)	12 (0.6)
21 = (9,10)	5 (0.1)	12 (0.9)		

Table 4.7: Computing $F(1800) = 0.6675061884$ for Example 4.3.

Routine	Full Decomposition		10,000 Sets Decomposed				
	Sets	Time	Bounds		Time	$ U $	$P\{U\}$
MCDA-I							
-Feasible 1	21,148	76.78	0.66744115	0.66752595	40.66	3840	.00009
-Feasible 2	59,252	130.68	0.66412553	0.66934024	25.91	2362	.00522
-Infeasible 1	> 500,000		0.01365559	0.90549725	64.88	2259	.89184
-Infeasible 2	> 500,000		0.01060540	0.90549840	51.06	2283	.89489
-Hybrid	> 500,000		0.62086991	0.69095869	33.41	2564	.07009
MCDA-II							
-Feasible 1	> 500,000		0.51748501	0.69914193	25.06	4258	.18166
-Feasible 2	> 500,000		0.46837277	0.80181865	19.38	4000	.33340
-Infeasible 1	335,712	783.19	0.59588386	0.68162250	26.38	3350	.08574
-Infeasible 2	> 500,000		0.43849046	0.69493580	17.28	3367	.25640
-Hybrid	> 500,000		0.52844341	0.72242710	19.09	3251	.19398

Table 4.8: Bounding $F(2000)$ for Example 4.3.

Routine	10,000 Sets Decomposed				
	Bounds		Time	$ U $	$P\{U\}$
MCDA-I					
-Feasible 1	0.91922537	0.97045163	40.16	4751	.05123
-Feasible 2	0.91259465	0.97513833	27.85	4219	.06254
-Infeasible 1	0.05918410	0.99954251	31.44	2678	.94036
-Infeasible 2	0.00790706	0.99955266	24.57	2794	.99165
-Hybrid	0.81577920	0.99071067	33.47	3294	.17493
MCDA-II					
-Feasible 1	0.79366799	0.97677949	26.34	4643	.18311
-Feasible 2	0.78470934	0.99022890	18.59	4591	.20552
-Infeasible 1	0.51684255	0.9838851	27.62	4715	.46704
-Infeasible 2	0.29276291	0.98467936	18.41	4379	.69192
-Hybrid	0.77839503	0.98117291	19.35	3760	.20280

Table 4.9: Bounding the cdf $F(d)$ for Example 4.3.

d	$F_l(d)$	$F_u(d)$	d	$F_l(d)$	$F_u(d)$
1600	0.1390231	0.1390231	2150	0.8421683	0.9998394
1650	0.3164801	0.3274079	2200	0.8623073	0.9999785
1700	0.3775094	0.4164037	2250	0.8903911	0.9999958
1750	0.4869543	0.5755966	2300	0.9221442	0.9999993
1800	0.5640803	0.7176557	2350	0.9459363	0.9999999
1850	0.6247240	0.8149941	2400	0.9685088	0.9999999
1900	0.6932091	0.9173447	2450	0.9813164	0.9999999
1950	0.7380265	0.9631111	2500	0.9906127	1.0000000
2000	0.7690676	0.9893741	2550	0.9965654	1.0000000
2050	0.7918757	0.9961165	2600	0.9983520	1.0000000
2100	0.8171786	0.9931391	2650	0.9998800	1.0000000

Table 4.10: Input Probability Distribution for Example 4.4.

Arc	(Capacity, Cost)		(Probability)	
1 = (1,2)	(70,15) (0.5)	(73,12) (0.3)	(94,10) (0.2)	
2 = (1,3)	(25,15) (0.5)	(35,10) (0.4)	(42,7) (0.1)	
3 = (1,4)	(42,13) (0.5)	(48,10) (0.3)	(61,8) (0.2)	
4 = (2,5)	(26,17) (0.4)	(31,10) (0.3)	(55,8) (0.2)	(88,6) (0.1)
5 = (2,6)	(58,10) (0.4)	(70,8) (0.3)	(95,7) (0.3)	
6 = (3,2)	(15,15) (0.6)	(73,9) (0.4)		
7 = (3,7)	(65,16) (0.5)	(74,10) (0.4)	(75,8) (0.1)	
8 = (3,8)	(59,12) (0.6)	(72,10) (0.3)	(98,8) (0.1)	
9 = (4,3)	(21,18) (0.3)	(32,13) (0.3)	(85,12) (0.2)	(98,10) (0.2)
10 = (4,9)	(89,11) (0.7)	(96,7) (0.3)		
11 = (5,7)	(32,14) (0.6)	(48,12) (0.2)	(67,8) (0.2)	
12 = (5,10)	(63,17) (0.5)	(99,15) (0.5)		
13 = (6,3)	(66,20) (0.8)	(85,12) (0.1)	(98,7) (0.1)	
14 = (6,5)	(6,12) (0.4)	(15,10) (0.3)	(39,8) (0.2)	(58,5) (0.1)
15 = (6,7)	(2,8) (0.6)	(48,7) (0.4)		
16 = (7,8)	(16,12) (0.3)	(18,11) (0.3)	(40,8) (0.2)	(52,5) (0.2)
17 = (7,10)	(16,14) (0.5)	(34,10) (0.4)	(71,6) (0.1)	
18 = (8,4)	(90,13) (0.5)	(96,9) (0.5)		
19 = (8,9)	(17,14) (0.4)	(49,12) (0.4)	(53,10) (0.1)	(65,8) (0.1)
20 = (9,7)	(3,12) (0.5)	(30,9) (0.4)	(50,7) (0.1)	
21 = (9,10)	(6,12) (0.5)	(12,10) (0.3)	(54,8) (0.1)	(66,5) (0.1)

Table 4.11: Computing $F(1900) = 0.36696615$ for Example 4.4.

Routine	Full Decomposition		500 Sets Decomposed				
	Intervals	Time	Bounds		Time	$ \mathcal{U} $	$P\{\mathcal{U}\}$
MCDA-I							
-Feasible 1	559	3.81	0.36692783	0.36699223	3.38	32	.000065
-Feasible 2	1497	6.16	0.36368208	0.37200554	2.43	135	.008323
-Infeasible 1	3179	22.09	0.34150064	0.37240647	6.22	248	.030906
-Infeasible 2	3179	22.20	0.34150064	0.37240647	6.34	248	.030906
-Hybrid	> 100,000		0.23197761	0.39739952	13.32	1662	.165422
MCDA-II							
-Feasible 1	> 100,000		0.29763540	0.41543235	6.84	2274	.1178
-Feasible 2	> 100,000		0.24929387	0.65997558	5.46	4079	.4107
-Infeasible 1	21,469	124.15	0.30655235	0.41347403	5.56	1114	.10695
-Infeasible 2	> 100,000		0.24629346	0.42723007	4.18	1413	.18094
-Hybrid	59,857	224.80	0.32552317	0.40912128	6.06	1786	.083598

Table 4.12: Computing $F(3000) = 0.99908320$ for Example 4.4.

Routine	Full Decomposition		500 Sets Decomposed				
	Intervals	Time	Bounds		Time	$ \mathcal{U} $	$P\{\mathcal{U}\}$
MCDA-I							
-Feasible 1	2908	24.06	0.99875743	0.99930461	4.84	445	.000547
-Feasible 2	6127	30.78	0.99869394	0.99940407	4.79	412	.000710
-Infeasible 1	8455	42.03	0.97500671	0.99931205	5.50	405	.024305
-Infeasible 2	8455	42.72	0.97500671	0.99931205	5.45	405	.024305
-Hybrid	71,439	299.40	0.95753799	0.99949807	8.20	709	.041960
MCDA-II							
-Feasible 1	> 100,000		0.99793109	0.99958227	5.41	850	.001651
-Feasible 2	> 100,000		0.99789501	0.99970247	4.59	827	.001808
-Infeasible 1	19,711	108.85	0.99794290	0.99950216	6.44	1340	.001559
-Infeasible 2	> 100,000		0.83254224	0.99997934	5.34	3351	.167440
-Hybrid	55,659	215.15	0.99809057	0.99945495	5.87	1600	.001364

Table 4.13: Partial listing of the cdf $F(d)$ for Example 4.4.

d	$F(d)$	d	$F(d)$
1600	0.0240000	2150	0.7852381
1650	0.1285435	2200	0.8358933
1700	0.1665479	2250	0.8707846
1750	0.2019579	2300	0.9006702
1800	0.2400117	2350	0.9241992
1850	0.2834110	2400	0.9469183
1900	0.3669662	2500	0.9722436
1950	0.4770354	2600	0.9821508
2000	0.5725981	2700	0.9887615
2050	0.6538413	3000	0.9990832
2100	0.7296389	3500	0.9999979

Table 4.14: Input Probability Distribution for Example 4.5.

Arc	(Capacity, Cost) (Probability)			
1 = (1,2)	(70,15) (0.7)	(73,12) (0.2)	(84,10) (0.1)	
2 = (1,3)	(25,15) (0.9)	(35,10) (0.05)	(42,7) (0.05)	
3 = (1,4)	(42,13) (0.8)	(48,10) (0.1)	(51,8) (0.1)	
4 = (2,5)	(26,12) (0.6)	(31,10) (0.2)	(45,8) (0.1)	(50,6) (0.1)
5 = (2,6)	(58,10) (0.8)	(70,8) (0.1)	(85,7) (0.1)	
6 = (3,2)	(15,15) (0.9)	(23,9) (0.1)		
7 = (3,7)	(65,16) (0.7)	(74,10) (0.2)	(75,8) (0.1)	
8 = (3,8)	(59,12) (0.8)	(65,10) (0.1)	(78,8) (0.1)	
9 = (4,3)	(21,18) (0.6)	(32,13) (0.2)	(45,12) (0.1)	(50,10) (0.1)
10 = (4,9)	(89,11) (0.8)	(96,7) (0.2)		
11 = (5,7)	(32,14) (0.7)	(48,12) (0.2)	(57,8) (0.1)	
12 = (5,10)	(63,17) (0.8)	(79,15) (0.2)		
13 = (6,3)	(66,18) (0.8)	(75,12) (0.1)	(88,7) (0.1)	
14 = (6,5)	(6,12) (0.7)	(15,10) (0.1)	(19,8) (0.1)	(28,5) (0.1)
15 = (6,7)	(2,8) (0.9)	(8,7) (0.1)		
16 = (7,8)	(16,12) (0.6)	(18,11) (0.2)	(22,8) (0.1)	(35,5) (0.1)
17 = (7,10)	(16,14) (0.8)	(24,10) (0.1)	(31,6) (0.1)	
18 = (8,4)	(90,13) (0.8)	(96,9) (0.2)		
19 = (8,9)	(17,14) (0.7)	(29,12) (0.1)	(33,10) (0.1)	(45,8) (0.1)
20 = (9,7)	(3,12) (0.8)	(10,9) (0.1)	(15,7) (0.1)	
21 = (9,10)	(6,12) (0.6)	(12,10) (0.2)	(24,8) (0.1)	(26,5) (0.1)

methods. Finally, a partial listing of the distribution $F(d)$ is given in Table 4.17. Full decomposition considered 448,729 intervals. The mean optimal flow cost, as computed from this distribution, is 1689.7094 with standard deviation 135.01.

4.5 Conclusions

We conclude this chapter by making some observations about the use of partitioning routines to solve the stochastic minimum cost flow problem. Several points deserve mentioning at this time.

Table 4.15: Computing $F(1650) = 0.62234033$ for Example 4.5.

Routine	Full Decomposition		500 Sets Decomposed				
	Intervals	Time	Bounds		Time	$ \mathcal{U} $	$P\{\mathcal{U}\}$
MCDA-I							
-Feasible 1	15	0.47					
-Feasible 2	30	0.40					
-Infeasible 1	30	0.59					
-Infeasible 2	30	0.56					
-Hybrid	3,514	14.56	0.61302130	0.62792051	5.13	156	.014899
MCDA-II							
-Feasible 1	775	3.53	0.62234028	0.62234033	2.60	15	5.2E-8
-Feasible 2	> 100,000		0.53768333	0.76617651	5.12	3055	.22849
-Infeasible 1	28	0.53					
-Infeasible 2	6011	19.09	0.62233192	0.62234056	2.06	1046	8.6E-6
-Hybrid	1094	5.13	0.62232657	0.62234094	3.16	47	1.4E-5

Table 4.16: Computing $F(2000) = 0.95451624$ for Example 4.5.

Routine	Full Decomposition		500 Sets Decomposed				
	Intervals	Time	Bounds		Time	$ \mathcal{U} $	$P\{\mathcal{U}\}$
MCDA-I							
-Feasible 1	1440	10.68	0.95414827	0.95469526	4.56	321	.000547
-Feasible 2	3601	16.56	0.95362469	0.95519911	3.41	282	.001574
-Infeasible 1	2832	21.31	0.95074918	0.95516215	5.84	190	.004413
-Infeasible 2	2832	20.63	0.95074918	0.95516215	5.44	190	.004413
-Hybrid	> 100,000		0.85767863	0.96448481	8.32	1037	.106806
MCDA-II							
-Feasible 1	> 100,000		0.94053428	0.95723072	5.54	1491	.016696
-Feasible 2	> 100,000		0.93344046	0.96810572	4.87	2283	.034665
-Infeasible 1	19,346	116.28	0.94785747	0.95657733	5.97	1175	.008720
-Infeasible 2	> 100,000		0.81657072	0.96065541	4.28	1977	.144085
-Hybrid	61,931	264.53	0.94789470	0.95614263	6.35	1260	.008248

Table 4.17: Partial listing of cdf $F(d)$ for Example 4.5.

d	$F(d)$	d	$F(d)$
1600	0.2469600	1850	0.8618947
1625	0.4921901	1900	0.9090322
1650	0.6223403	2000	0.9545162
1675	0.6355670	2100	0.9916873
1700	0.6593318	2200	0.9982173
1725	0.7007062	2300	0.9997193
1750	0.7725537	2400	0.9999767
1775	0.8127293	2500	0.9999996
1800	0.8348261	2600	1.0000000
1825	0.8431940		

- It is clear that, compared to the success of the MST routines in Chapter 3, the results of the present chapter do not stand out as being phenomenal.
- Since each arc now has two associated random variables the size of the state space is effectively *squared*. Taking this into consideration, most of the decomposition results here perform well in terms of only evaluating a small fraction of the points in the state space.
- It seems clear that our conjectures in Section 3.5 remain valid for MCF problems. Since these problems do not have a matroid structure, we did not expect the kind of performance we saw in Chapter 3. In particular, the lack of an easy way to determine a new minimum cost flow when arc costs or capacities changed really hurt. We were left either using a cutoff state that was not deep enough or reoptimizing over and over to try to improve such a point.
- In Chapter 3, the simpler routine STDA-II (which did not derive infeasible sets by finding limiting indices) was more effective than the more complete, sophisticated STDA-I. Here we have the opposite case. The inability to derive easy cut-off states necessitates the derivation of infeasible sets.
- Here it is reasonable only to compare the hybrid routines to the GSD routines that are based on the same cutoff state derivations (i.e., MCDA-I or MCDA-II). In all cases, as predicted in Section 2.5, these did not perform as well as the better GSD routines.

In summary, the algorithms given here, especially in light of the above comments, have performed very well. The astronomically large state spaces and lack of special matroid structure certainly hindered their effectiveness somewhat. But considering the impossibly hard task of evaluating these measures in the absence of such routines (by enumeration or crude Monte Carlo simulation), the algorithms of this chapter must be called successful.

Finally, there are several important network optimization problems that can be cast as minimum cost network flow problems. Furthermore, independence among components in these problems translates into independence in the present setting so that application of our methods makes sense. However, we point out that we would expect for GSD-type algorithms tailored to exploit the special structure of one of these problems to out-perform the more general algorithms given here when applied to the same problem. Indeed, this has been seen to be the case [2].

CHAPTER 5

Conclusions and Directions for Further Study

In the preceding chapters we have attempted to accomplish several goals. First, we have tried to improve the state of knowledge concerning solution methodologies that use iterative partitioning of state spaces. Secondly, we have endeavored to adapt these procedures to create efficient means for studying minimum spanning trees in stochastic networks. Finally, we have attempted to likewise address certain probabilistic minimum cost network flow problems. We discuss here the extent to which we have succeeded in these aims, and propose directions for further study.

In Chapter 2, our task was to investigate a computational approach that has been used with some success in addressing other problems involving stochastic network systems. This approach, best described as a variant of factoring, was shown to be applicable to a rather broad class of stochastic systems. Our description of the General State Space Decomposition (GSD) algorithm was intended in part to make a contribution by being complete and flexible enough to make this powerful technique easily adaptable to all problems in this class of systems.

In addition, our description was designed to facilitate some theoretical investigations. Specifically, we posed some questions concerning the partitioning step used in GSD in hopes that their answers might shed some light on certain performance issues. We essentially sought to find the best approach to removing feasible and infeasible states from an undetermined interval. Our analysis was based on the

intuitive notion that performance would suffer whenever an interval was split into many small pieces instead of fewer, larger pieces. Our conclusion, that any departure from GSD would result in an unnecessary proliferation of filed undetermined sets, led us to predict that, in general, the better GSD routines should out-perform arbitrary decomposition schemes applied to the same problem. This conjecture was supported by virtually every application studied in Chapters 3 and 4.

Nevertheless, there are some performance issues that remain to be investigated. These difficult issues deal with finding the best GSD scheme for a given problem. Why, for example, do certain cutoff state derivations perform well on some instances and poorly on others? Given a number of alternative cutoff state derivations, is there a way to determine which derivation will perform best for a particular problem instance? These questions are difficult because of the complex interplay of different factors that determine a GSD algorithm's ultimate effectiveness for a given problem, but are nonetheless worthy of further exploration.

The subject of Chapter 3 was the application of GSD to problems related to minimum spanning trees in stochastic networks. Both problems addressed there were shown to be computationally hard, a heretofore unproved result for the criticality index problem. Despite this, we found that these algorithms were very effective. We hypothesized that this was due in part to the matroid structure of the MST problem. The fact that the algorithms in Chapter 4 (applied to a problem with no matroid structure) performed worse seemed to support this conjecture.

The routine used to compute criticality indices stood out as being particularly efficient. Motivated by this fact, we proposed in Section 3.6 a heuristic algorithm for determining the membership of the most probable minimum spanning tree. We believe this to be worthy of study and intend to investigate this application in the future. In addition, from a more theoretical standpoint, it might be fruitful to look for underlying structural reasons for the success of GSD applied to this problem. Such information might help us to predict with improved confidence the kinds of problems that will readily admit solution by GSD and to better tailor

specific decomposition methods to those problems.

In Chapter 4 we applied partitioning algorithms to stochastic minimum cost network flow problems. These problems, too, were shown to be #P-hard. This gave us cause for concern since in this setting each arc has two associated random variables, resulting in state spaces that are larger than those for corresponding MST problems. Furthermore, as mentioned above, the MCF problem has no matroid structure. Consistent with this expectation, computational performance was not as remarkable as that which we saw in Chapter 3. However, we were still able to get usable computational results in most cases, and these results continued to confirm the results of Section 2.5. Particularly effective were the algorithms applied to the case of dependent arc costs and capacities. These applications were also important because they illustrated a new flexibility of GSD and represented, as far as we know, the first time vector-valued variables were used in a decomposition approach of this kind.

However, for large problems it would likely be worthwhile to write variance-reducing Monte Carlo estimation routines of the form given in Section 2.7. This investment in additional effort would allow us to get a good set of initial bounds for P^F via GSD, and then use the remaining undetermined intervals to efficiently produce an unbiased estimate of P^F .

This thesis has made a number of contributions. First, it has shown several important stochastic network problems to be computationally hard, and then has presented efficient means for solving these problems exactly. In addition, it has advanced the understanding of state space partitioning methods. In doing so, it has made these methods more accessible and has drawn some strong conclusions about the soundness of using GSD-type routines instead of arbitrary decomposition schemes. Finally, it has proposed some areas in which further gains might be made with regards to this powerful computational technique.

Bibliography

- [1] Alexopoulos, C. (1991). Computing criticality indices of arcs and the mean maximum flow value in networks with discrete random capacities, Technical Report, School of Industrial and Systems Engineering, Georgia Institute of Technology, submitted for publication.
- [2] Alexopoulos, C. (1992). State space partitioning methods for stochastic shortest path problems, Technical Report, School of Industrial and Systems Engineering, Georgia Institute of Technology.
- [3] Alexopoulos, C. and G.S. Fishman (1991). Characterizing stochastic flow networks using the Monte Carlo method, *Networks*, **21**, 775-798.
- [4] Alexopoulos, C. and G.S. Fishman (1991). Sensitivity analysis in stochastic flow networks using the Monte Carlo Method, *Networks*, forthcoming.
- [5] Bailey, M.P. (1990). Minimization on stochastic matroids, NPS-55-90-01, Naval Postgraduate School, Monterrey, California.
- [6] Ball, M.O. (1987). Computational complexity of network reliability analysis: An overview, *IEEE Transactions on Reliability* **35**, 230-239.
- [7] Ball, M.O., Colbourn, C.J. and J.S. Provan (1992). Network Reliability (Submitted for publication).
- [8] Barlow, R.E. and Proschan, F. (1981). *Statistical Theory of Reliability and Life Testing*, Holt, Reinhart and Winston.

- [9] Bertsekas, D.P. and P. Tseng (1988) The RELAX codes for linear minimum cost network flow problems, *Annals of Operations Research*, **7**, 125-190.
- [10] Bertsimas, D.J. (1990). The probabilistic minimum spanning tree problem, *Networks*, **20**, 245-175.
- [11] Bertsimas, D.J. and G. van Rysin (1990). An asymptotic determination of the minimum spanning tree and minimum matching constants in geometrical probability, *Operations Research Letter*, **9**, 223-231.
- [12] Camerini, P.M., Galbiati, G. and F. Maffioli (1988). Algorithms for finding optimum trees: description, use and evaluation, *Annals of Operations Research*, **13**, 265-397.
- [13] Chen, H. and J. Zhou (1991). Reliability optimization in generalized stochastic-flow networks, *IEEE Transactions on Reliability*, **40**, 92-97.
- [14] Corea, G.A. and V.G. Kulkarni (1990). Minimum cost routing on stochastic networks, *Operations Research*, **38**, 527-536.
- [15] Doulliez, P. and E. Jamouille (1972). Transportation networks with random arc capacities, *R.A.I.R.O.*, **3**, 45-60.
- [16] Fishman, G.S. and T. Shaw (1989). Evaluating reliability of stochastic flow networks, *Probability in the Engineering and Informational Sciences*, **3**, 493-509.
- [17] Frieze, A.M. (1985). On the value of a random minimum spanning tree problem, *Discrete Applied Mathematics*, **10**, 47-56.
- [18] Gilbert, E.M. (1965). Random minimal trees, *J. SIAM*, **13**, 376-387.
- [19] Grimmett, G.R. and D.J.A. Welsh (1982). Flow in networks with random capacities, *Stochastics*, **7**, 205-229.

- [20] Hagstrom, J.N. (1983). Using the decomposition-tree of a network in reliability computation, *IEEE Transactions on Reliability*, **32**, 71-78.
- [21] Hagstrom, J.N. and P. Kumar (1984). Reliability computation on a probabilistic network with path length criterion, Technical Report, University of Illinois, Chicago.
- [22] Jain, A. and J.W. Mamer (1988) Approximations for the random minimal spanning tree with applications to network provisioning, *Operations Research*, **36**, 575-584.
- [23] Kulkarni, V.G. (1986). Minimum spanning trees in undirected networks with exponentially distributed arc weights, *Networks*, **16**, 111-124.
- [24] Lawler, E.L. (1976). *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston.
- [25] Le, K.V. and V.O.K. Li (1989). Modeling and analysis of systems with multimode components and dependent failures, *IEEE Transactions on Reliability*, **29**, 68-75.
- [26] Leuker, G.S. (1981). Optimisation Problems on graphs with independent edge weights, *SIAM Journal of Computing*, **10**, 338-351.
- [27] Nagamochi, H. and T. Ibaraki (1992). On Onaga's upper bound on the mean value of probabilistic maximum flows, *IEEE Transactions on Reliability*, **41**, 225-229.
- [28] Nemhauser, G.L. and L.A. Wolsey (1988). *Integer and Combinatorial Optimization*, Wiley-Interscience.
- [29] Papadimitriou, C. and K. Stieglitz (1982). *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall.

- [30] Prékopa, A. and E. Boros (1991). On the existence of a feasible flow in a stochastic transshipment network, *Operations Research*, **39**, 119-129.
- [31] Steele, J.M. (1987) Growth rates on Frieze's $\zeta(3)$ limit for lengths of minimal spanning trees, *Discrete Applied Mathematics*, **18**, 99-103.

VITA

Captain Jay Alan Jacobson was born to James Edmund and Diana Sue Jacobson on September 15, 1961, in Birmingham, Alabama. He attended Huffman High School. Upon graduating in 1979, he entered the University of Alabama under a four-year Air Force ROTC scholarship to study mathematics. In 1983 he was awarded the Bachelor of Science degree (*magna cum laude*) and was commissioned an officer in the United States Air Force.

Air Force assignments have included serving as Chief of Airman Modeling at the Air Force Military Personnel Center at Randolph AFB, Texas, and being an instructor in the Department of Mathematical Sciences at the United States Air Force Academy in Colorado.

He was awarded a Master of Science degree in operations research from Stanford University in 1988. He began studies in the School of Industrial and Systems Engineering at the Georgia Institute of Technology in 1989, and was awarded the degree of Doctor of Philosophy in 1993.

He is married to the former Jennifer Karen Swann of Northport, Alabama. They have two children. John Christian was born in 1985 at Fort Sam Houston, Texas, and William Alan was born in 1989 at the Air Force Academy.